

Curso HTML 5

Ministerio
de Educación, Cultura
y Deporte

COLECCIÓN AULA MENTOR

SERIE DISEÑO WEB

CamSdw

Curso HTML 5

Diseño y Autoedición

Autor

Francisco Javier Pascual Romera

Coordinación pedagógica

Rocío de la Bandera Berlanga

Gemma Villegas Díaz

Edición y maquetación de contenidos

Natalia Guaita Hernández

Diseño gráfico e imagen

Natalia Guaita Hernández



MINISTERIO
DE EDUCACIÓN, CULTURA
Y DEPORTE

Edita:

© SECRETARÍA GENERAL TÉCNICA
Subdirección General
de Documentación y Publicaciones

NIPO: 030-15-1897
ISBN: 978-84-369-5639-9

ÍNDICE

	Pág.
1.- Generalidades HTML5	2
1.1. Definición de html5	3
1.2. Lenguajes de marcas	5
1.2.1. ¿Qué son?	5
1.2.2. Tipos	6
1.2.3. Características	7
1.2.4. Ejemplos de lenguajes de marcas	7
1.3. Historia de html5	10
1.4. Compatibilidad de navegadores con html5	11
1.5. Ventajas de html5	13
1.6. Novedades	14
1.7. Conclusión	16
2.- Manejo básico de HTML	18
2.1. Relación de html5 con html4 y xml	19
2.1.1. Especificaciones oficiales	20
2.2. Etiquetas	21
2.2.1. Definición	21
2.2.2. Atributos	22
2.3. Elementos html	23
2.4. Estructura de un documento html	27
2.4.1. Elemento <html>	27
2.4.2. Cabecera de un documento html. <head>	31
2.4.2.1. Html5 y los caracteres especiales castellanos (ñ, á, ü, etc.)	37
2.4.3. Cuerpo de un documento html. <body>	39
2.5. Validación de una página web	41
2.6. Comentarios	43
2.7. Divisiones, párrafos y agrupaciones	45
2.7.1. Ejercicios	50
2.8. Tablas	56
2.8.1. Operaciones con tablas	66
2.9. Listas	71
2.9.1. Anidamiento de listas	78
2.10. Enlaces	83

2.10.1.URL (localizador uniforme de recursos)	83
2.10.2.Etiqueta <a>	87
2.11. Imágenes.....	94
2.12. Iframes	101
3.-Los nuevos elementos semánticos HTML5.....	111
3.1. Estructura de una página html5	113
3.2. Elementos semánticos html5	115
3.2.1. Encabezados o títulos de secciones.....	116
3.2.2. Los nuevos elementos estructurales de HTML5.....	117
3.2.3. Otros nuevos elementos semánticos HTML5 relacionados con el aspecto.....	132
4. CSS es un lenguaje de estilo	142
4.1. Definición de CSS3.....	143
4.1.1. Ventajas de las hojas de estilo CSS	143
4.2. Incorporación de estilos a los documentos.....	145
4.3. Selectores.....	151
4.4. Herencia y cascada	152
4.4.1. Herencia.....	152
4.4.2 Cascada.....	157
4.5. Unidades de medida.....	160
4.5.1. Unidades Absolutas.....	160
4.5.2.Unidadesrelativas.....	161
4.5.3.Porcentajes.....	161
4.6. Modelo de cajas.....	162
4.6.1 Dimensiones.....	162
4.6.2. Propiedad “Sintaxis alternativa” (Shorthand)	165
4.6.3. Margen.....	166
4.6.3. Borde.....	167
4.6.4. Relleno.....	170
4.6.5 Fondos.....	172
4.7. Textos y fuentes.....	178
4.7.1.Propiedadesdetexto.....	178
4.7.2. Propiedades de las fuentes o tipos de letras	183
4.8. Visibilidad.....	189
4.9. Posicionamiento.....	208
4.9.1. Posicionamiento normal o estático.....	209
4.9.2. Posicionamiento relativo.....	209
4.9.3. Posicionamiento absoluto.....	216
4.9.4. Posicionamiento fijo.....	216
4.9.5. Solapamiento de elementos.....	
4.9.6. Posicionamiento flotante.....	230
4.10. Listas.....	240
4.10.1. Propiedades listas.....	240
4.11. Algunas innovaciones CSS3.....	254
4.11.1. CSS3 Borders.....	255
4.11.2. Backgrounds.....	257
4.11.3. Gradientes.....	263
4.11.4. Flex o Flexbox.....	263

5. HTML5	308
5.1 Introducción.....	308
5.2. Reglas básicas.....	310
5.2.1. Tipos de controles.....	310
5.3. Elemento <form>.....	312
5.4. Elemento <input>.....	320
5.4.1. Controles básicos <input>.....	323
5.4.2. Controles nuevos HTML5 <input>.....	329
5.5. Otros elementos de formulario	332
5.6. Ejemplo de construcción de un formulario con estilos.....	341
6. Los nuevos elementos semánticos HTML5.....	350
6.1. Introducción.....	350
6.2. Vídeo.....	351
6.2.1. Formatos de vídeo.....	351
6.2.2. Elemento <video>.....	352
6.2.3. Elemento <source>.....	356
6.2.4. Elemento <track>.....	357
6.3. Audio.....	360
6.3.1. Formatos de audio.....	360
6.3.2. Elemento <audio>.....	361
6.4. Elementos <object> y <embed>.....	363
7. Los nuevos elementos semánticos HTML5.....	372
7.1. Generalidades.....	373
7.2. Elementos de definición de scripts.....	374
7.3. Incorporación de scripts a un documento html.....	377
7.4. Programación inicial.....	384
7.4.1. Reglas sintácticas básicas.....	384
7.4.2. Variables.....	384
7.4.3. Operadores	386
7.4.4. Estructuras de control	388
7.4.5. Funciones.....	392
7.5. DOM.....	394
7.6. Eventos.....	415
8 HTML5 avanzado.....	438
8.1. Introducción.....	439
8.2. Geolocation.....	441
8.3. DRAG and DROP.....	444
8.4. Canvas.....	450
8.4.1. Coordenadas canvas.....	454
8.4.2. Dibujando una línea recta.....	454
8.4.3. Dibujando un círculo.....	457
8.4.4. Gradientes o difuminados.....	461
8.4.5. Texto.....	466
8.4.6. Imágenes.....	470

BLOQUE 1:

GENERALIDADES HTML5

1. GENERALIDADES HTML5

Este bloque trata de dar una idea general de lo que es HTML5, en qué se basa, cuáles son sus predecesores y cuáles son sus ventajas sobre otras versiones de HTML.

Para entender el funcionamiento básico de HTML5 es importante conocer los fundamentos de los lenguajes de marcas y su evolución desde sus primeras versiones allá por los años 60 ya que nos va a permitir introducirnos en el mundo de la creación de páginas web.

HTML5 simplifica y mejora la creación de documentos HTML, eliminando etiquetas obsoletas, apoyándose en las hojas de estilo para mejorar su aspecto, en javascript para ganar dinamismo y funcionalidad y en sus nuevos elementos para no tener que usar programas externos o para crear una estructura más coherente dentro de una página web.

Ante las innovaciones que aporta HTML5 es interesante saber el grado de compatibilidad que tienen los navegadores a la hora de tratar con los nuevos elementos HTML5 ya que esto nos permitirá utilizar esos elementos para que no haya problema a la hora de verlos en cualquier navegador.

También hay que ver que HTML5 va más allá de lo que es un lenguaje de marcas y verlo como una agrupación de especificaciones que están marcando el desarrollo web.

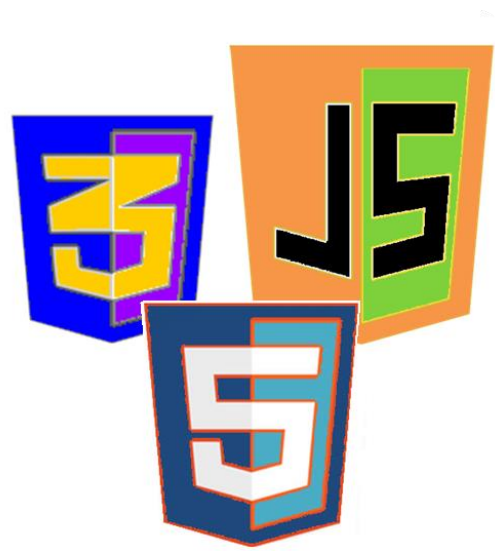
En definitiva, tal y como lo describe el consorcio W3C, **HTML5 es el futuro del contenido.**

1.1. DEFINICIÓN DE HTML5

HTML5 es lo que se denomina un **lenguaje de marcas** que, combinado con **CSS3**, **JavaScript** y algunas tecnologías de diseño, permite la creación de páginas web.

Es la quinta revisión importante del lenguaje básico de Internet denominado HTML.

Cuando creamos una página web debemos saber que esta no es ni más ni menos que un documento de texto compuesto por contenidos o textos que manejan los elementos HTML, los cuales se rigen por las normas del lenguaje HTML, y que se visualiza a través de una aplicación denominada navegador (Google Chrome, Internet Explorer, Ópera, Mozilla Firefox, etc.) que interpreta su contenido HTML.



El lenguaje HTML es un estándar cuyas normas, a partir de su versión 3.0, son definidas por un organismo sin ánimo de lucro denominado World Wide Web Consortium, más conocido como **W3C**.

El propio W3C define el lenguaje HTML como "un lenguaje reconocido universalmente y que permite publicar información de forma global".

Esta definición debería implicar que una misma página web se vería igual a través de un navegador u otro pero esto no ocurre ya que los diferentes navegadores interpretan de forma muy desigual el lenguaje HTML de una misma página web dejando la visualización del contenido más en manos del navegador que de su propio diseño.

El estándar HTML5 tratará en el futuro de paliar este problema y todos los navegadores lo deberán interpretar de la misma manera.



W3C España - W3C España x

www.w3c.es/estandares/

Vistas: escritorio móvil impresión

ESTÁNDARES PARTICIPAR UNIRSE SOBRE EL W3C

W3C España » Estándares

ESTÁNDARES

El W3C desarrolla especificaciones técnicas y directrices a través de un proceso que ha sido diseñado para maximizar el consenso sobre el contenido de un informe técnico, de forma que se pueda asegurar la alta calidad técnica y editorial, así como obtener un mayor apoyo desde el W3C y desde la comunidad en general.

El W3C ofrece varias **vistas** de sus especificaciones, incluyendo:

- por tema de la tecnología [\[inglés\]](#) (como por ejemplo, "todas las especificaciones relacionadas con HTML")
- por estado [\[inglés\]](#) (esta es una aproximación a la vista "clásica" de los informes técnicos (TR), con publicaciones recientes al principio y después todas las especificaciones agrupadas por su estado, desde Recomendaciones a borradores de trabajo)
- por fecha [\[inglés\]](#) (las más recientes al principio)
- por grupo [\[inglés\]](#) (quien desarrolla la especificación)

Si estás aprendiendo sobre tecnología Web, puede que quieras comenzar por la introducción de más abajo, y seguir los enlaces para más detalle.

Diseño y Aplicaciones Web [\[inglés\]](#)

Diseño y Aplicaciones Web incluye a los estándares para la construcción y representación de las páginas Web, incluyendo HTML5, CSS, SVG, Ajax y otras tecnologías para las Aplicaciones Web ("WebApps"). Esta sección también incluye información sobre cómo hacer páginas accesibles para personas con discapacidades (WCAG), aplicar internacionalización y trabajar sobre dispositivos móviles.

ENLACES RÁPIDOS

- Todas las Especificaciones [\[inglés\]](#)
- Recientes [\[inglés\]](#)
- Suscribirse [\[inglés\]](#)
- Preguntas frecuentes (FAQ) [\[inglés\]](#)
- Sobre los Estándares [\[inglés\]](#)

USAR LOS ESTÁNDARES

- Validador W3C
- Chuleta W3C [\[inglés\]](#)
- Tutoriales [\[inglés\]](#)
- Traducciones [\[inglés\]](#)

WEB PARA TODO EL MUNDO

1.2. LENGUAJES DE MARCAS

1.2.1. ¿Qué son?

Cuando escribimos un documento y queremos resaltar un carácter, palabra o frase lo que hacemos normalmente es subrayarlo, escribirlo más fuerte o usar caracteres más grandes para que destaque del resto del texto, es decir, estamos aplicando un **formato** concreto al texto con el objeto de distinguirlo de la información que está a su alrededor.

Utilizando el párrafo anterior a modo de ejemplo, la palabra **formato**, destaca del resto por queremos llamar la atención sobre él, por eso la hemos puesto en negrita y con un tipo de letra mayor que la del resto del documento, es decir, le hemos dado un determinado formato.

Un lenguaje de marcas define un conjunto de marcas o etiquetas y la estructura que debe tener un documento para poder aplicar dichas marcas.

```
<!DOCTYPE html>
<html>
<head><title>Listas anidadas</title></head>
<body>
<ol>
  <li>Provincias, comarcas y pueblos </li>
    <li>Soria
      <ul>
        <li>Pinares
```

Una "marca" o "etiqueta" es una señal colocada dentro de un documento que delimita una parte de él a la que aplica un determinado formato en base a la función que tiene asignada dicha marca en el lenguaje.

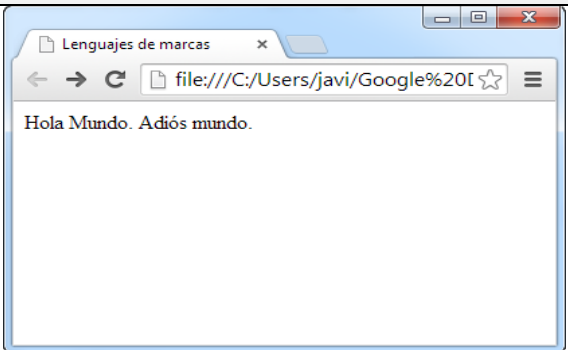
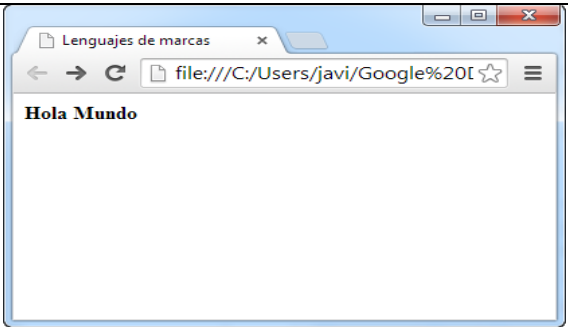
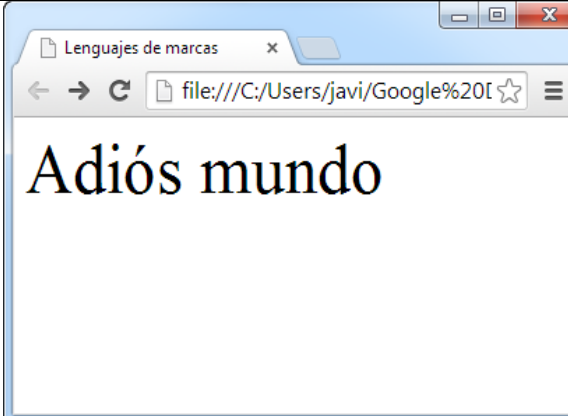


Un lenguaje de marcas no es un lenguaje de programación



Un lenguaje de marcas se puede combinar con un lenguaje de programación para aportar nuevas funcionalidades y dar más calidad a la página web

Ejemplos:

Lenguaje de marcas	Texto que se visualiza
Hola Mundo. Adiós mundo.	
<code> Hola Mundo.</code>	
<code> Adiós mundo</code>	

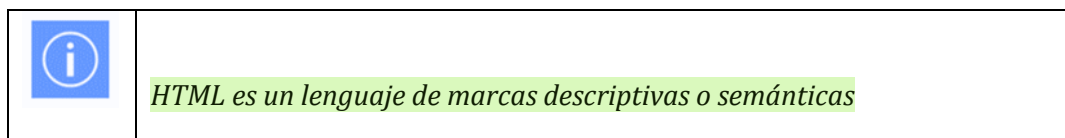
En el primer ejemplo se ve el texto tal y como se escribe ya que dicho texto no está dentro de una etiqueta. En el segundo ejemplo, la etiqueta `<b>` pone en negrita el texto que hay hasta la etiqueta de cierre `</b>` y, en el tercer ejemplo, `<font size="7">` pone el texto hasta la etiqueta de cierre `</font>` a un mayor tamaño

1.2.2. Tipos

Se suelen diferenciar tres tipos de lenguajes de marcas según el tipo de marcas que definen aunque en un mismo documento puede haber marcas de los tres tipos. Estos lenguajes son:

- *Lenguaje de marcas de presentación*: Es aquel cuyas marcas definen el formato del texto que engloban lo que permite modificar el aspecto estético de un documento aunque no sea suficiente para el procesamiento de información.

- *Lenguaje de marcas de procedimiento*: Es aquel cuyas marcas son una serie de órdenes normalmente de tipo tipográfico para la presentación de textos.
- *Lenguaje de marcas descriptivas o semánticas*: Es aquel cuyas marcas identifican bloques dentro del documento y describen su contenido sin entrar en aspectos de formato.



1.2.3. Características

- *Texto Plano*

Los archivos de texto plano son aquellos que están compuestos únicamente por caracteres de texto, a diferencia de los archivos binarios que pueden contener imágenes, sonido, archivos comprimidos, programas compilados, etc.

Una de las principales ventajas de los archivos de texto plano es que pueden ser interpretados directamente por un simple editor de texto, a diferencia de los binarios que necesitan software específico (visores, descompresores, compiladores, etc.) Esta característica hace que los documentos sean independientes del sistema operativo o programa con el que fueron creados, esto facilita la interoperabilidad, que constituye una importante ventaja para el intercambio de información en Internet.

- *Integración*

Las instrucciones de marcado se mezclan con el propio contenido. El texto entre las marcas es el propio contenido del documento.

- *Independencia del dispositivo*

Inicialmente los lenguajes de marcas se idearon para visualizar documentos de texto, pero progresivamente se han empezado a utilizar en muchas otras áreas como gráficos vectoriales, sindicación de contenidos, notación científica, interfaces de usuario, síntesis de voz, etc.

- *Flexibilidad*

Los lenguajes de marcas se pueden combinar en el mismo archivo con otros lenguajes, como HTML con PHP y JavaScript. Incluso hay etiquetas específicas para ello como es `<script>`.

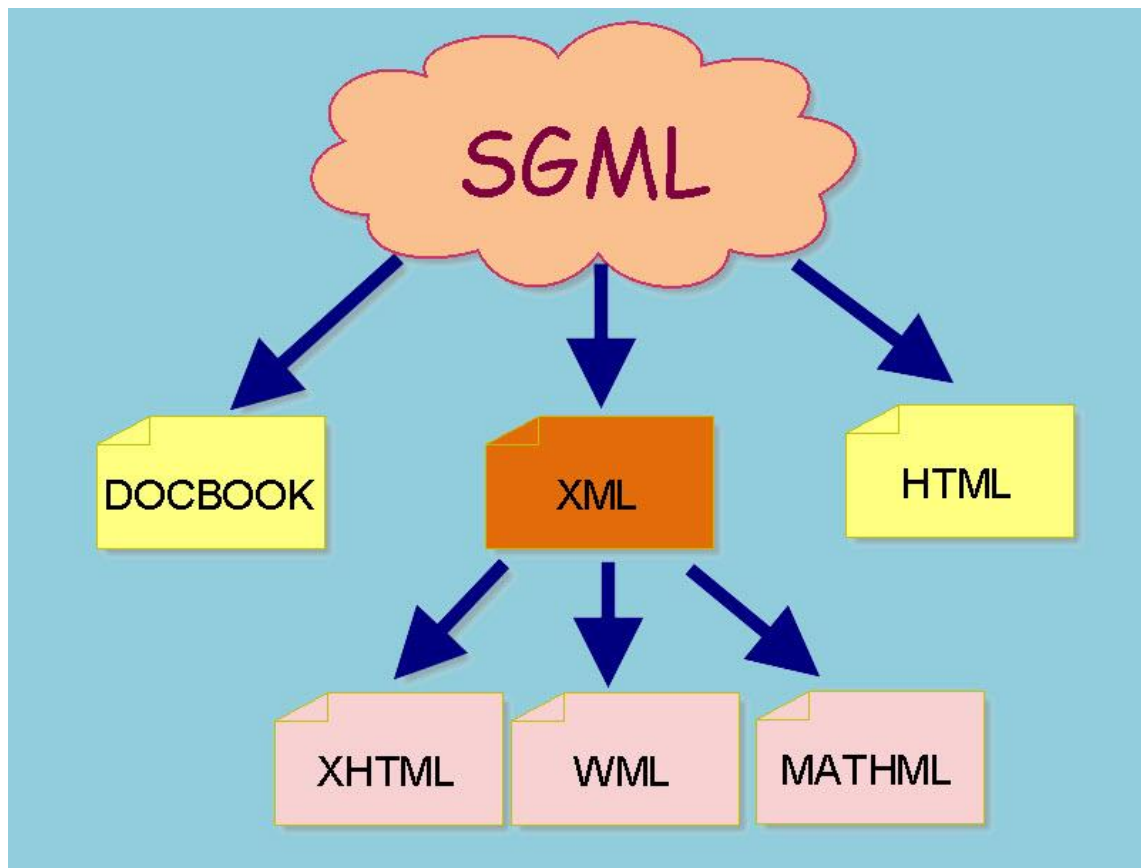
1.2.4. Ejemplos de lenguajes de marcas

- *SGML* (Standard Generalized Markup Language): Al igual que su antecesor GML, SGML o ISO 8879 es un lenguaje de marcas genérico, contiene las normas para generar otros lenguajes de marcas sin tener en cuenta su codificación interna por lo que se puede denominar que es un metalenguaje del que se derivan otros lenguajes más especializados y menos generalistas. Esto se hace posible a través del uso de una definición de tipo de documento (DTD, Document Type Definition), en la que se

especifican los elementos, atributos y su orden que se emplean en los documentos que cumplen un determinado lenguaje.

	<i>El organismo W3C se encarga de mantener las DTD de los lenguajes de marcas más usados como las versiones de HTML a excepción de HTML5 que no tiene.</i>
---	--

- **HTML:** Basado en SGML, HTML o Hypertextual Markup Language (lenguaje de marcas de hipertexto), es un estándar para la creación de páginas web que define una estructura básica (todos los documentos HTML tienen una cabecera <HEAD> y un cuerpo <BODY>) y un código genérico, denominado HTML, que indica los elementos y su funcionalidad de los mismos para todas las versiones aunque cada una de ellas tiene su propio DTD con los componentes y normas que las diferencian.
- **XML:** Subconjunto de SGML adaptado a las necesidades de intercambio de datos entre diversas aplicaciones, XML o Extensible Markup Language (lenguajes de marcas extensible) no es propiamente un lenguaje de marcas (no se usa para hacer páginas web) sino que es un metalenguaje en el que se basan otros lenguajes de marcas ya que se pueden definir etiquetas para determinadas funciones.
- **XHTML:** Es nueva adaptación de HTML con arreglo a las normas XML lo que obliga a que los documentos XHTML, Extensible HyperText Markup Language, deban estar “bien formados”, es decir, deben cumplir estrictamente la especificación XML, cosa que no sucede con los documentos HTML, lo que hace que los navegadores los visualicen más fidedignamente.



1.3. HISTORIA DE HTML5

IBM desarrolló en 1986 un lenguaje de marcas denominado GML (Generalized Markup Language) que pretendía resolver los problemas de compatibilidad que había entre aplicaciones debido a que cada aplicación utilizaba sus propias etiquetas o marcas para describir sus diferentes elementos y era prácticamente imposible el intercambio de información.

GML usa marcas genéricas que permiten distinguir el contenido del documento del aspecto y estructura del mismo.

Ese mismo año GML pasó a ser de la organización ISO convirtiéndose en el lenguaje SGML (ISO 8879), Standard Generalized Markup Language, con las características de ser un lenguaje de código abierto y de libre disposición. A partir de este momento se puede hablar de SGML como metalenguaje, es decir, lenguaje que sirve de base a otros lenguajes, en este caso, de marcas debido a que tiene un grupo de elementos y unas reglas que marcan la utilización de los elementos anteriores y sus atributos.

AÑO	Lenguaje
-1986	GML
1986	SGML
1991	HTML
1996	CSS, JavaScript
1998	HTML4
2000	XHTML 1.0
2003	XHTML 2.0
2009	HTML5

En 1991 nace HTML (HiperText Markup Language) a partir de SGML con la finalidad de ser el lenguaje que se emplee para la elaboración de páginas web. En sus versiones finales como HTML 4.01 se agregan hojas de estilo CSS para los aspectos estéticos y javascript para darle más dinamismo.

Después de HTML 4.01, el consorcio W3C, comunidad internacional que desarrolla estándares como HTML, desarrolló el lenguaje XHTML 1.0 que es mucho más rígido que HTML (es obligatorio escribir en minúsculas las etiquetas) pero que intentaba que todos los navegadores vieran igual el mismo documento HTML, cosa que no pasa como HTML 4.01.

A continuación se desarrolló el lenguaje XHTML 2.0 que modifica y mejora el lenguaje XHTML 1.0

Seguidamente un grupo de fabricantes de navegadores y desarrolladores web crearon una nueva especificación de HTML superior a HTML 4.01 orientada a crear un nuevo tipo de aplicaciones web denominada HTML5 y se abandonó el desarrollo de XHTML 2.0.

Es conveniente usar las versiones más modernas de navegadores que reconozcan los nuevos elementos HTML5 para poder sacar rendimiento a este lenguaje.

1.4. Compatibilidad de navegadores con HTML5

Los pasos que seguimos cuando queremos ver una página web creada por nosotros son:

- Creación del documento HTML con un editor normal o con un editor web siguiendo las normas y usando los elementos de la versión HTML que queramos utilizar en dicho documento.
- Visualización del documento con un navegador web como puede ser Google Chrome, Mozilla Firefox o Internet Explorer, entre otros.

Pero cuál es nuestra sorpresa al ver que la misma página web no se ve igual en distintos navegadores. Es más, si nos referimos al nuevo estándar HTML5, vemos que hay elementos que ni tan siquiera los reconocen algunos navegadores.

Ante esta perspectiva lo único que podemos hacer es consultar en

páginas de referencia en Internet el grado de compatibilidad con HTML5 que tiene un navegador, es decir, que nos indique los elementos y características HTML5 que soporta dicho navegador.



La razón de hacer esto se debe a que debemos conocer cuáles son los elementos que se reconocen en los navegadores más utilizados por los usuarios y usarlos en nuestras páginas. Esto nos dará la certeza de que nuestra página web se visualizará sin problemas en cualquier navegador.

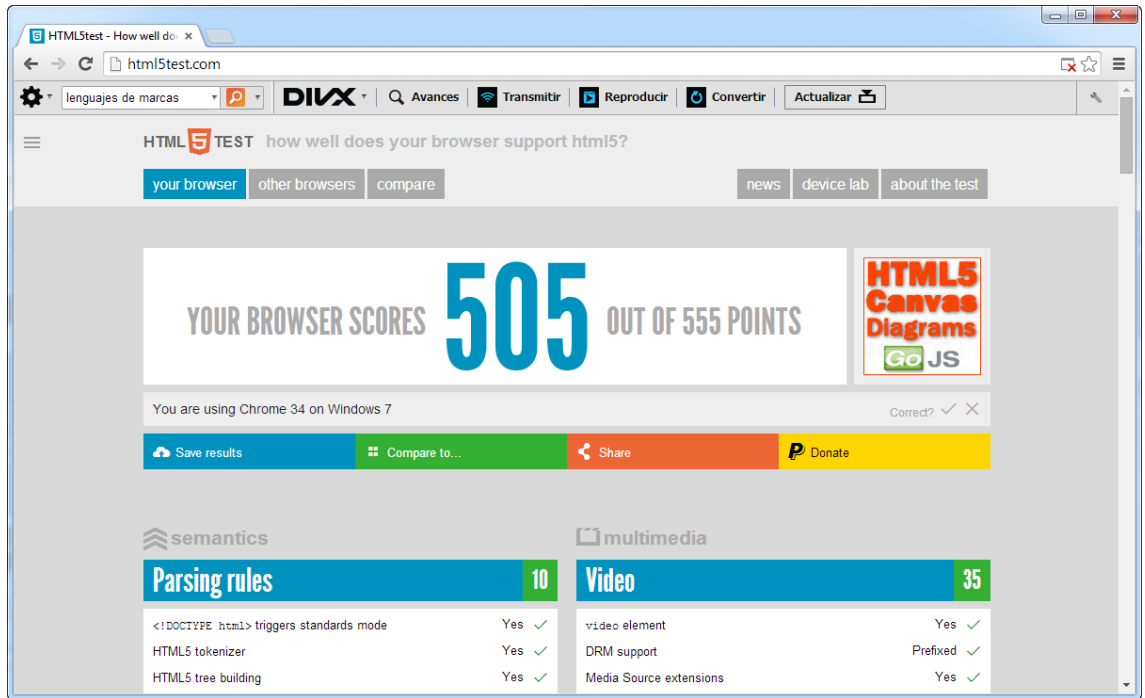
Otro problema añadido es la controversia sobre cuáles son los navegadores más utilizados en la red ya que hay estadísticas muy dispares sobre el tema pero podemos optar por informarnos sobre las últimas versiones de Google Chrome, Internet Explorer, Mozilla Firefox y Ópera.

Una de estas páginas de consulta es <http://html5test.com> que permite conocer la compatibilidad del navegador con el que se visualiza dicha página y tiene comparativas entre los principales navegadores que se usan incluidos los de Tablet y móviles.



El navegador Google Chrome en cualquiera de sus versiones es el que mejor reconoce los nuevos elementos y características HTML5 además de ser, posiblemente, el más utilizado por los usuarios.

Ejercicio: Comprueba el grado de compatibilidad con html5 que tienen tus navegadores y compáralos a través de la página <http://html5test.com>



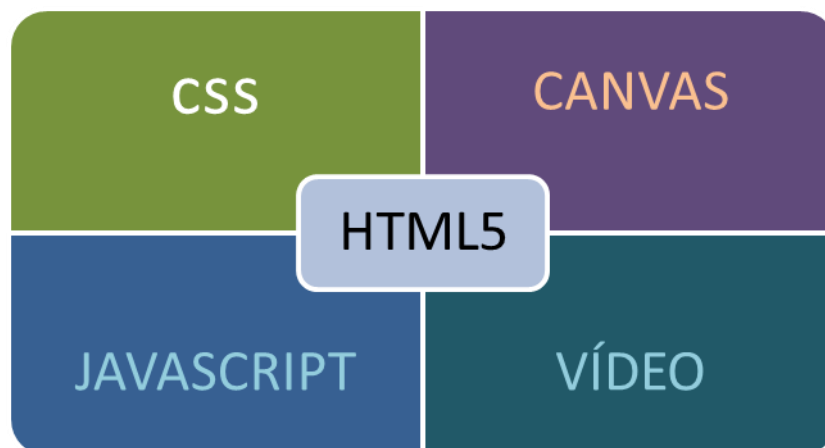
The screenshot shows the HTML5Test website interface. At the top, it says "HTML5 TEST how well does your browser support html5?". Below this, there are tabs for "your browser", "other browsers", and "compare". The main display shows "YOUR BROWSER SCORES 505 OUT OF 555 POINTS". A message indicates "You are using Chrome 34 on Windows 7". There are buttons for "Save results", "Compare to...", "Share", and "Donate". Below the score, there are two sections: "semantics" and "multimedia". The "semantics" section has a sub-header "Parsing rules" with a score of 10, and a table showing compatibility for "<!DOCTYPE html> triggers standards mode", "HTML5 tokenizer", and "HTML5 tree building", all marked "Yes". The "multimedia" section has a sub-header "Video" with a score of 35, and a table showing compatibility for "video element", "DRM support", and "Media Source extensions", all marked "Yes".

Category	Feature	Result
semantics	Parsing rules 10	
	<!DOCTYPE html> triggers standards mode	Yes ✓
	HTML5 tokenizer	Yes ✓
multimedia	Video 35	
	video element	Yes ✓
	DRM support	Prefixed ✓
	Media Source extensions	Yes ✓

Vídeo: compatibilidad.mp4

1.5. VENTAJAS DE HTML5

- Mayor compatibilidad entre los navegadores ya que el aspecto de una página web en uno u otro navegador no difiere tanto como cuando se visualizaban páginas HTML 4 o XHTML.
- Mayor rapidez a la hora de cargar la página debido a la simplificación de las etiquetas.
- Nuevas etiquetas que permiten crear una estructura bastante intuitiva de un documento HTML.
- Inserción de audio y vídeo de forma directa sin necesidad de emplear ningún plug-in.
- Permite la geolocalización de un usuario.
- Incorporación de más efectos visuales a través de la etiqueta canvas.
- El aspecto estético del documento se basa casi por completo en el uso de hojas de estilo CSS eliminando o reduciendo a la mínima expresión el uso de etiquetas de formato.
- La incorporación de nuevas capacidades Javascript.

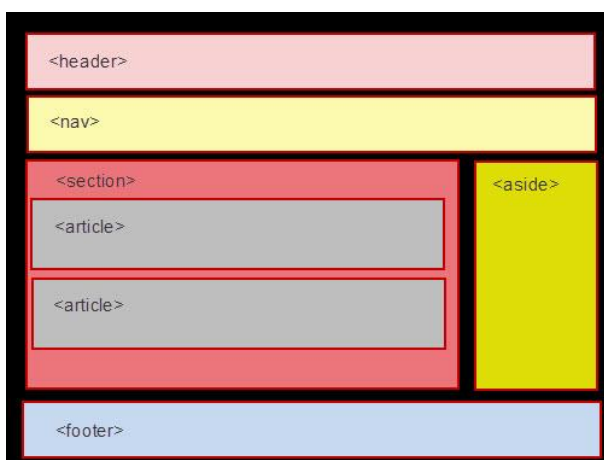


1.6. NOVEDADES

- **Nuevas etiquetas estructurales**

La estructura normal de una página web tiene un armazón bastante concreto basado en una cabecera, menús, cuerpos, pies, etc. Con HTML5 se usan etiquetas específicas (denominadas semánticas porque aportan información relevante) para definir todos estos elementos. Su uso no es obligatorio pero sí bastante descriptivo ya que un vistazo sobre el código html nos indica lo que hay en cada sección. Las etiquetas html4 `<div>` y `<span>` empleadas para definir partes de una página prácticamente no se usan ya que `<header>`, `<nav>`, `<section>` y otras etiquetas nuevas de html5 son las encargadas de definir las partes de una página web. Estos son los elementos:

- **Section:** Representa un bloque general dentro de un documento, es decir, un bloque de una misma temática. Puede contener subsecciones o artículos.
- **article** representa un contenido específico en un documento.
- **aside** representa un contenido complementario o muy poco relacionado con el resto de la página.
- **header** representa la cabecera de una página html o de una sección.
- **footer** representa el pie de una página o sección, con información acerca de la misma que va al final de la página o sección.
- **nav** zona de navegación con enlaces dispuestos de diferentes maneras.
- **hgroup:** Sirve para el agrupamiento de titulares.



- **Nuevas etiquetas para video, audio y tratamiento de gráficos**

Las etiquetas `<video>` y `<audio>` permiten la inserción de audio y vídeo en una página html5 de manera nativa sin depender de los plug-ins o código que necesitan los navegadores para saber qué hacer con un determinado archivo.

La etiqueta `<canvas>` permite trabajar con gráficos y animaciones.

La etiqueta `<embed>` sirve para manejar contenido incrustado pero no nativo, sino ejecutado por plugins como el de Flash.

- **Nuevas APIS**

Una interfaz de programación de aplicaciones (API) es un conjunto de instrucciones, funciones y normas de programación para acceder a una aplicación de software. Uno de los objetivos principales de una API es el de normalizar la comunicación entre aplicaciones.

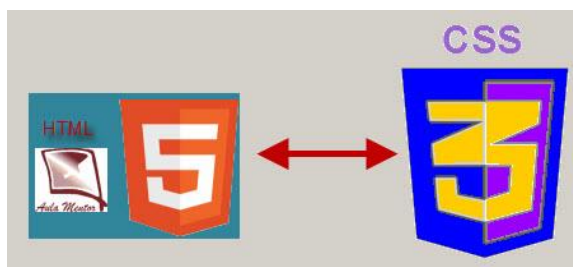
Las APIs son un aspecto muy importante dentro del entorno de HTML5 y hay una serie de ellas que conviene conocer, como Web Storage, Microdata o Geolocation (que permite conocer el punto geográfico desde el cual se conecta el navegador a Internet), entre otras y, en el futuro, se van a incorporar varias APIs nuevas que

van a permitir trabajar con Bases de datos en local, drag & drop (arrastrar y soltar) lo cual va a hacer de html5 un lenguaje mucho más potente.

- **Relación muy fuerte con hojas de estilos CSS3**

Los aspectos estéticos y de formato de una página html5 recaen casi completamente sobre el uso de Hojas de Estilo en Cascada (CSS) lo que facilita separar contenidos de formatos de presentación. De hecho el estándar 3.0 de CSS ha ido evolucionando en paralelo con html5, tanto que hay algunos componentes CSS3 que se confunden con elementos html5.

Por esta razón la inmensa mayoría de las etiquetas html 4 de formato pasan a quedar obsoletas y a no ser utilizadas con lo que se simplifica el número de etiquetas HTML5.



- **Mejores formularios**

Un formulario es una sección compuesta de controles a través de los cuales se recopila información introducida por el usuario para luego ser utilizada.

HTML5 añade características nuevas para la realización de formularios mejorando sustancialmente los de la anterior versión de HTML o HTML4.01.

La principal novedad es la validación de datos por parte del navegador o del lado cliente lo que no hará necesario el empleo de programas auxiliares y se simplifica notablemente tanto la creación como el uso de formularios. Para ello incluye nuevos campos de entrada de datos y atributos para otros controles.

	No todos los navegadores permiten la validación del lado cliente. En el futuro si lo deberían de poder hacer.
---	--

Un ejemplo que constata todo lo anterior es el elemento input el cual ha sido simplificado en su sintaxis pero ampliado en funcionalidades.

1.7. CONCLUSIÓN

HTML5 representa el futuro de la creación de páginas web pero debemos saber que, aunque simplifique y estructure mejor el código HTML, agregue nuevos atributos y etiquetas y elimine aquellas que están obsoletas, no es solo una nueva versión de un lenguaje de marcas HTML sino una agrupación de diversas especificaciones que están influyendo en el desarrollo web.

A su vez, HTML5 pretende proporcionar una plataforma con la que desarrollar aplicaciones web muy parecidas a las aplicaciones de escritorio, es decir, cercano a lo que se denomina Web 2.0. (uso de aplicaciones de web en vez de locales y que la propia web sea la plataforma donde se integran estas aplicaciones). Para ello se están creando APIs que permitan trabajar con cualquiera de los elementos de la página sin necesidad de usar programas adicionales lo que debería generar un estándar que todos los navegadores lo reconocieran de la misma manera y desaparecieran los problemas de compatibilidad entre las funcionalidades de este estándar y los navegadores.

Por eso el consorcio W3C habla de HTML5 como El futuro del contenido web.



BLOQUE 2:

MANEJO BÁSICO DE HTML

2. MANEJO BÁSICO DE HTML

En este tema vamos a trabajar en los aspectos fundamentales de la construcción de un documento HTML5, que no es otra cosa que un archivo de texto, para lo cual debemos conocer las peculiaridades de este lenguaje de marcas como son su estructura, elementos etiquetas, etc. que tienen bastante en común con las anteriores versiones de HTML por lo que esta primera aproximación al diseño web nos va a llevar a familiarizarnos con los componentes elementales de cualquier versión HTML y sus reglas para la construcción de páginas web. Veremos la estructura de un documento HTML, lo que son los elementos HTML alrededor de los cuales gira toda creación HTML, y su componente más importante: las etiquetas o marcas y, por último, crearemos nuestras primeras páginas web.

2.1. RELACIÓN DE HTML5 CON HTML4 y XML

Tanto HTML5 como HTML4, XML y XHTML son lenguajes de marcas lo que implica el uso de etiquetas para darle un determinado aspecto a un texto o ejecutar una determinada acción. Hay que recordar que estos cuatro lenguajes de marcas tienen bastantes estructuras comunes lo que permite que, si se conoce cualquiera de ellos, se puede aprender la sintaxis de los demás sin gran esfuerzo aunque difieran en la uso de unas normas de escritura más o menos estrictas.

El precedente de estos lenguajes de marcas es el estándar internacional SGML (Standard Generalized Markup Language) que define la estructura de diferentes tipos de documentos basándose en la relación lógica de sus partes.

XML y HTML son lenguajes muy diferentes. Su sintaxis es similar, aunque cada uno fue diseñado para cumplir distintas funciones.

El XML (eXtensive Markup Language) es un lenguaje que fue concebido para describir información. Su función principal es ayudarnos a organizar contenidos y eso hace que los documentos XML sean portables hacia diferentes tipos de aplicaciones y que los navegadores los interpreten mejor.

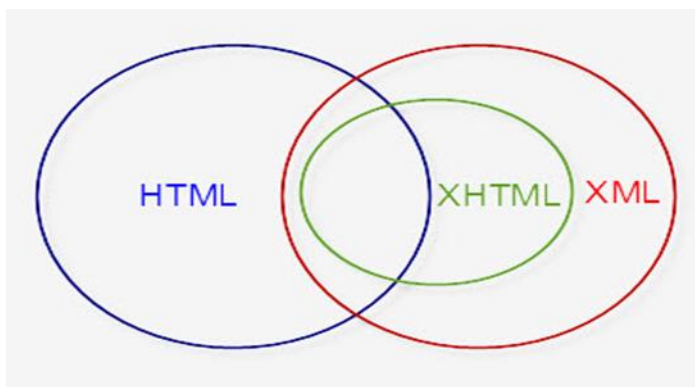
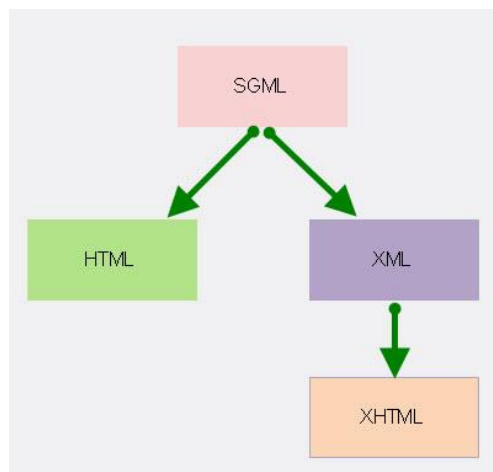
Un dato importante es que al hablar de XML hablamos de documentos bien formados (well formed), es decir, aquellos que deben cumplir estrictamente tanto las normas sintácticas como semánticas, mientras que en HTML podemos ser desordenados y no tiene la rigidez de XML.

HTML (HyperText Markup Language), por otro lado, ha sido concebido para mostrar información, determinar cómo actúa y qué hace. Su función radica en ayudarnos a darle formato a los diversos contenidos de una página.

XHTML es muy parecido a HTML ya que es una adaptación de HTML al lenguaje XML lo que significa que XHTML tiene las mismas funcionalidades de HTML pero cumple las especificaciones más estrictas de XML. Su aparición intentaba paliar un gran problema del lenguaje HTML y es que los navegadores representaban de distinta forma la misma página html. XHTML, al ser más estricto, permite a los diferentes navegadores representar prácticamente de la misma manera una misma página html.

XHTML se considera descendiente más de XML que de HTML.

Resumiendo, el XML sirve para describir información y el HTML sirve para darle formato y presentarla a través de un navegador. O sea que el XML no ha sido nunca un reemplazo de HTML sino un complemento que sirve para manejar la información separada del formato.



Para finalizar HTML5 es la última versión de HTML cuya sintaxis es compatible con HTML4 y con XHTML.. Este incorpora elementos innovadores, emplea hojas de estilo en cascada (CSS) para los efectos estéticos de aspecto, no reconoce algunos de los elementos de HTML4 e implementa nuevas funcionalidades a través de APIs Javascript.

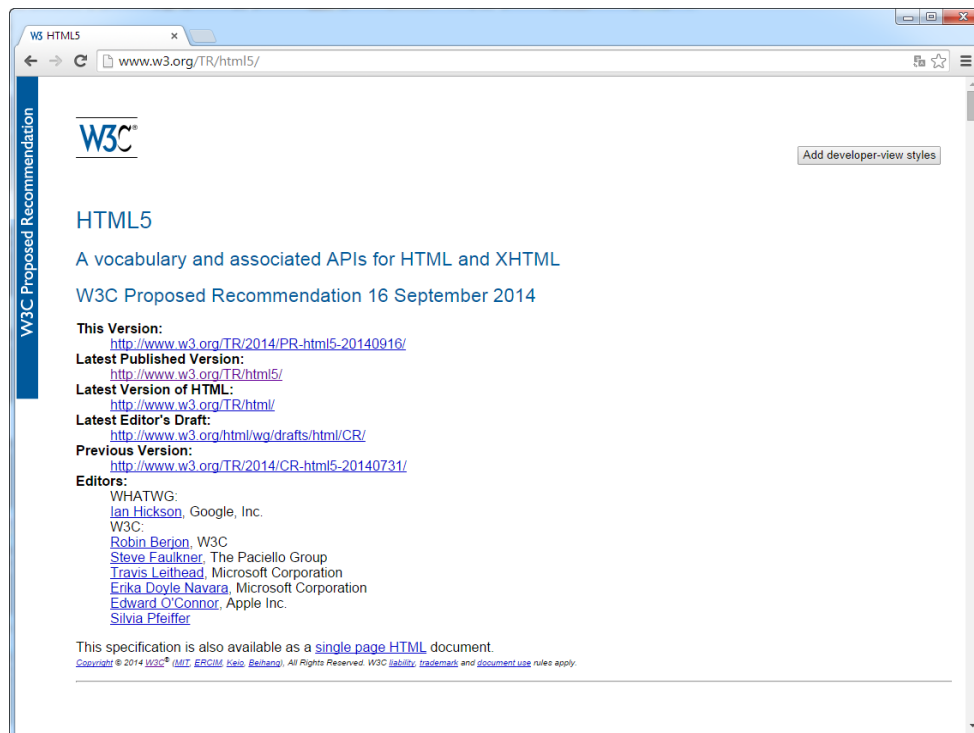
2.1.1. Especificaciones oficiales

El organismo W3C (*World Wide Web Consortium*) elabora las normas que deben seguir los usuarios de cualquiera de esos lenguajes anteriores para que no haya problemas a la hora de reconocer el tipo de documento que se ha creado.

Todos los lenguajes de marcas usados en Internet tienen unas especificaciones obligatorias de cumplir que están marcadas en documentos publicados por ese organismo.

En el caso de HTML5 podemos encontrar esa especificación en la dirección:

<http://www.w3.org/TR/html5/>

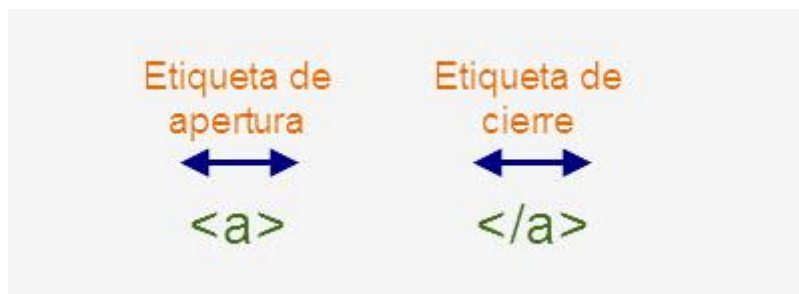


2.2. ETIQUETAS

2.2.1 Definición

Las etiquetas son textos incluidos entre los símbolos *menor que* "<" y *mayor que* ">" que marcan los diferentes elementos que integran una página html y les aplican características concretas a estos. Aunque hay versiones de html que permiten usar tanto mayúsculas como minúsculas para escribir las etiquetas, es muy conveniente hacerlo solo en minúsculas.

Existe normalmente una etiqueta de inicio y otra de fin (hay excepciones en las que solo se necesita la etiqueta de inicio). En HTML la etiqueta de fin es la misma que la de inicio añadiéndole al principio una barra inclinada "/" y afecta a todo lo que haya entre ambas ya sea texto u otras etiquetas.



En esta imagen se usa como ejemplo la etiqueta <a> que sirve para insertar un enlace o hipervínculo en una página html.

	<i>En todo lenguaje de marcas existe un número concreto de etiquetas con una sintaxis y semántica predefinida que desempeñan una función preestablecida de antemano</i>
--	--

	<i>Una etiqueta en HTML se puede escribir en mayúsculas o minúsculas indistintamente pero se recomienda escribirlas en minúsculas.</i>
--	---

HTML5 incorpora nuevas etiquetas y no soporta algunas de HTML 4.01

- Listado de etiquetas HTML5 (guías de referencia).
- Listado de etiquetas HTML 4.01 no soportadas en HTML5 (guías de referencia).

2.2.2. Atributos

Los atributos son aspectos que se pueden modificar del funcionamiento de una etiqueta y que se insertan en cualquier orden dentro de la etiqueta de inicio poniendo el nombre del atributo y su valor entre comillas (" ") separados por un signo *igual* (=)



En la imagen **href** es el atributo particular que modifica la etiqueta **<a>** cuyo valor es <http://www.mentor.mec.es> y que indica a donde apunta el hipervínculo, por lo que, la etiqueta completa es la referencia a la página principal del proyecto Mentor.

No todos los atributos se pueden utilizar en todas las etiquetas. Por ello, cada etiqueta define su propia lista de atributos disponibles. Además, cada atributo también indica el tipo de valor que se le puede asignar.

	Si el valor de un atributo no es válido, el navegador lo ignora
---	---

Aunque cada una de las etiquetas HTML define sus propios atributos, algunos de ellos son comunes a muchas o casi todas las etiquetas siendo su funcionalidad la misma independientemente de la etiqueta en la que se aplica. Los atributos comunes se dividen en tres grupos según su funcionalidad:

- **Atributos globales** se pueden utilizar prácticamente en todas las etiquetas HTML (ver apéndice/referencia). Los más comunes son los siguientes:

Atributo	Descripción
class	Especifica uno o más nombres de clase para un elemento (hojas de estilo CSS)
id	Identificativo del elemento.
style	Estilo que se aplica al elemento
title	Información extra sobre el elemento.

- **Atributos particulares:** Son aquellos que solo afectan a una etiqueta en concreto.

- **Atributos de eventos:** sólo se utilizan en las páginas web dinámicas creadas con JavaScript (Ver apéndice/referencia)

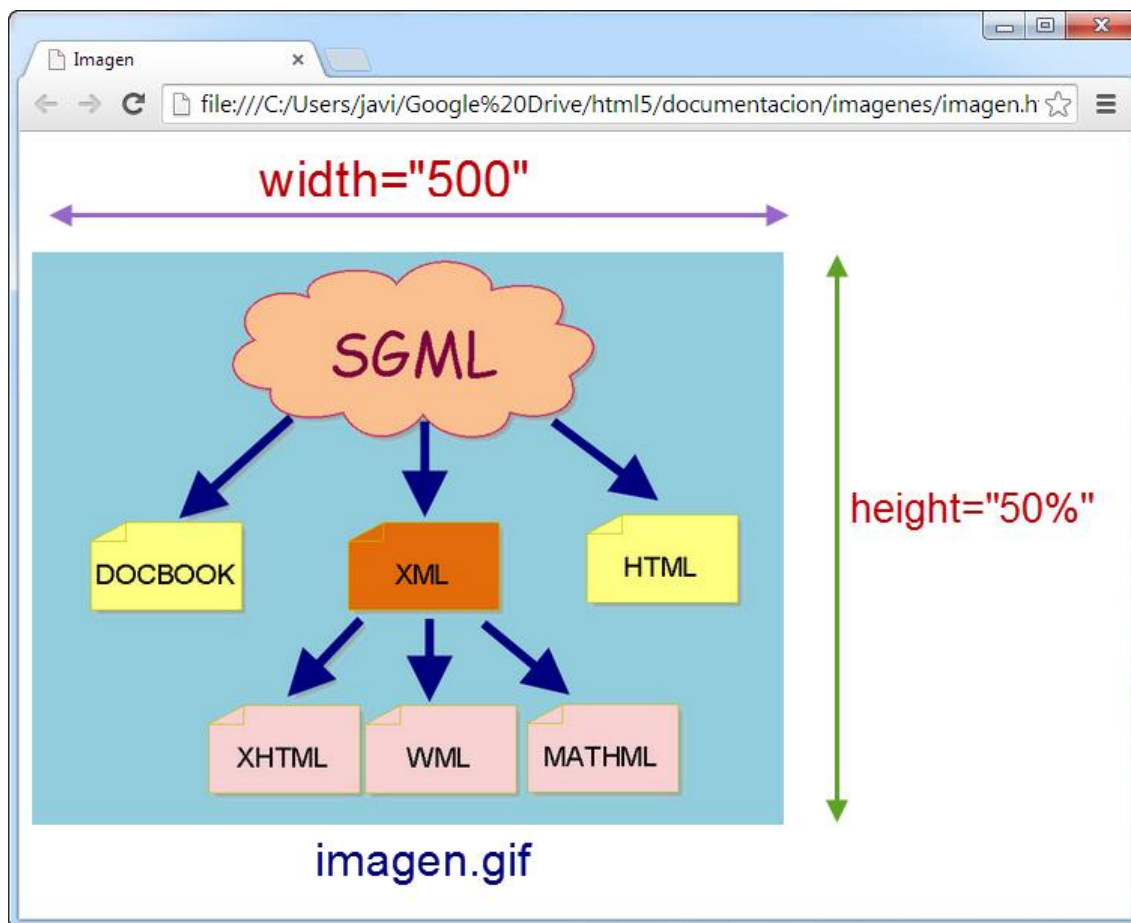
Ejemplo:

```

```

La etiqueta `img` permite la inserción de una imagen en el archivo `html`. En este caso el atributo `src` indica que archivo se va a visualizar, el atributo `width` indica que la imagen tendrá 500 pixels de anchura y `height` indica que ocupa el 50% del alto de la pantalla.

En la siguiente imagen se ve cómo quedaría el archivo `imagen.gif` dentro de una página web que lo visualiza.



2.3. Elementos HTML

Los elementos HTML son los auténticos componentes de una página web ya que marcan cada una de las partes de los documentos HTML.

	Un elemento HTML no es una etiqueta
---	-------------------------------------

Muchas veces se confunde el término “elemento” con el de “etiqueta” ya que se piensa que son dos cosas iguales pero un elemento html es más que una etiqueta ya que está formado normalmente por:

- Una etiqueta de apertura.
- Cero o más atributos en la etiqueta de apertura.
- Texto u otros elementos que son formateados con arreglo a la función de la etiqueta.
- Una etiqueta de cierre.



En el ejemplo anterior el elemento html está formado de la etiqueta `<a>` (indica que es un hipervínculo o enlace), el atributo **href** (indica la dirección hacia donde nos dirigirá el enlace) cuyo valor es **http://www.mentor.mec.es** (dirección del enlace), el contenido cuyo valor es **Enlace** (texto al cual se aplica la etiqueta html) y una etiqueta de cierre `</a>` que es igual a la de apertura pero con la barra /.

Algunos elementos no llevan etiqueta de cierre y son denominados elementos vacíos ya que no tienen contenido (un ejemplo es `<br>` que devuelve un retorno de carro).

	Un elemento vacío se puede cerrar dentro de la propia etiqueta de apertura. Ejemplo <code>
</code>
---	---

	<i>La mayoría de los elementos HTML se pueden anidar, es decir, un elemento HTML puede contener a otro u otros</i>
---	--

	<i>Un elemento HTML puede ser reconocido por un navegador pero puede que alguno de sus atributos no lo sea</i>
---	--

Por otra parte el lenguaje HTML clasifica a todos los elementos en tres grupos dependiendo de cómo ocupan el espacio en el documento. Estos grupos son de tipo bloque (block), de tipo en línea (inline) y los que no son ni en línea ni en bloque.

- **Elementos de tipo bloque (block):** Los elementos del tipo bloque tienen las siguientes características:
 - Forman un bloque y se posiciona dentro del documento html de forma vertical finalizado con un retorno de carro que lo separa de los demás elementos.
 - Ocupan toda la anchura que le proporciona el elemento en el cual está contenido aunque, usando hojas de estilos CSS, se puede aplicar una anchura fija.
 - Su altura va en función del contenido del elemento aunque, al igual que pasa con la anchura, se puede utilizar hojas de estilos CSS para aplicar una altura fija.
 - Puede contener otros elementos de tipo inline y/o block

Ejemplos de elementos de tipo bloque: <p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <div>, , , , <menú>, <div>, <pre>, <hr>, <blockquote>, <address>, <center>, <noframes>, <isindex>, <fieldset>, <table>, <form>

- **Elementos en línea (inline):** Los elementos de tipo en línea tienen las siguientes características
 - Se ajustan horizontalmente con los elementos que estén junto a él.
 - La anchura se define en base al contenido.
 - En el caso de permita contener otros elementos, estos deben ser de tipo inline, es decir, no pueden contener elementos de tipo block.

Ejemplos de elementos de tipo en línea: <a>,
, , <bdo>, <object>, <applet>, , <map>, <iframe>, <tt>, <i>, , <big>, <small>, <u>, <s>, <strike>, , <basefont>, , , <dfn>, <code>, <q>, <sub>, <sup>, <samp>, <kbd>, <var>, <cite>, <input>, <select>, <textarea>, <label>, <button>

- **Elementos que no son ni en línea ni bloque:** Son los elementos que no forman parte del flujo de la información porque son elementos subordinados a otros elementos.

Ejemplos de Elementos subordinados son

- Subordinados a table: caption tr th td thead tbody tfoot col colgroup
- Subordinados de listas: li dd dl

- Subordinados a select son: optgroup option
- subordinado a map: area
- subordinado a object: param
- subordinado a fieldset: legend

	Por medio de CSS se puede cambiar la forma de ver un elemento mediante las reglas display:block (visualizar el elemento en forma bloque) y display:inline (visualizar el elemento en forma en línea)
---	---

En el siguiente ejemplo se crean dos elementos de tipo bloque (párrafos <p>) y elementos en línea (enlaces <a>) y como se pueden anidar unos dentro de otros.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>En bloque y en línea</title></head>
5  <body>
6    <p>
7      Primera línea del párrafo <br>
8      Segunda línea del párrafo que es un poco más larga que la primera<br>
9      Tercera línea del párrafo
10   </p>
11
12   <a href="http://www.mentor.mec.es">enlace al proyecto mentor</a> pero puedo seguir
13   escribiendo al lado
14
15
16
17   <p>
18     Dentro de un bloque se pueden insertar otros elementos bloque:<br>
19
20     <ol>
21       <li> Las listas son de tipo bloque</li>
22       <li> Primer elemento </li>
23       <li> Segundo elemento </li>
24     </ol><br>
25
26     y de tipo en línea <a href="http://www.mentor.mec.es">enlace al proyecto mentor</a>
27     y sigo escribiendo
28   </p>
29
30 </body>
31 </html>
32

```

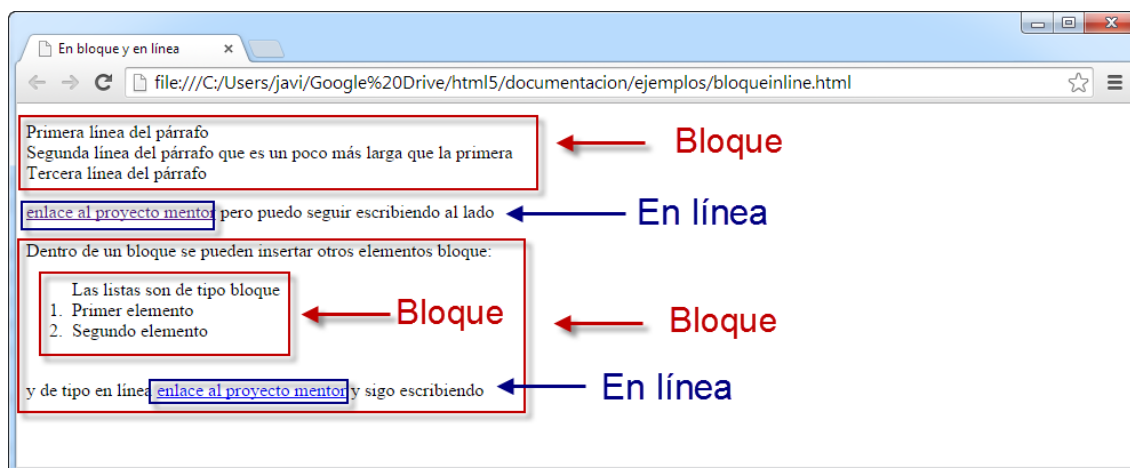
Bloque

En línea

Bloque

En línea

Visualización del ejemplo anterior en el navegador Chrome



2.4. ESTRUCTURA DE UN DOCUMENTO HTML

La estructura básica de una página web es la siguiente:

```
<html>

  <head>

    <title> Título de la página</title>

    Elementos de cabecera

    .

    .

  </head>

  <body>

    Elementos y contenido de la página

    .

    .

  </body>

</html>
```



La cabecera y el contenido de una página web está dentro del elemento `<html>` que comienza con la etiqueta `<html>` y finaliza con `</html>`

2.4.1 Elemento `<html>`

Etiqueta	<html>					
Descripción	Representa la raíz de un documento html					
Elemento	Bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	

Atributos	Globales
Atributos propios	manifest, xmlns
Diferencia entre html 4.01 y html5	El atributo manifest se ha añadido en HTML5

Atributos propios	Valor	Descripción
manifest 	URL 	Localización del archivo que sirve de caché o memoria temporal de la página web
xmlns 	http://www.w3.org/1999/xhtml/	Localización del espacio de nombres XML. Opcional en html5

 → Nuevo en html5

Su principal atributo global es lang que indica el idioma del contenido de los elementos del documento HTML.

En los siguientes ejemplos se indica que el lenguaje del contenido por defecto de los elementos de la página web está escrito en español (España).

```
<html lang="es">
```

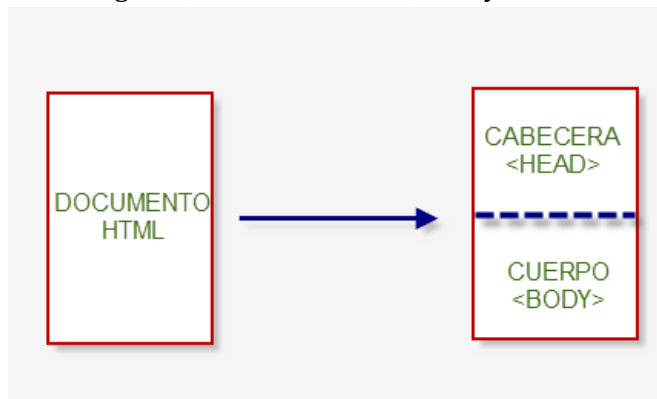
También se puede poner

```
<html lang="es-es">
```

<html> contiene otros dos elementos que son:

- <head> </head>: Es la cabecera de la página que contiene definiciones generales que afectan a todo el documento. Esta información no forma parte del contenido de la página.
Todos sus elementos son opcionales y se utilizarán en casos muy determinados. El elemento <title>, título de la página, no es estrictamente obligatorio pero siempre debe ponerse ya que permite poner un título en la parte superior de la ventana del navegador que de indicios del contenido de la página.

- `<body> </body>`: El elemento `<body>` define el cuerpo de un documento y contiene todos los elementos que se van a visualizar de la página web como textos, enlaces, imágenes, etc. En definitiva, `<body>` es el contenido del documento HTML.



La declaración HTML `<!DOCTYPE>`

Aunque la estructura anterior es perfectamente válida para cualquier versión de HTML o XML hay que reseñar que es muy importante indicarle al navegador el tipo de documento va a utilizar (si es HTML4, HTML5, etc.) a través de la declaración `<!DOCTYPE>`. Esta declaración siempre va antes que la etiqueta de inicio de documento `<html>`.

En html4, xml y xhtml dentro de la DOCTYPE va reflejada la dirección documento DTD (Definición del tipo de documento) que recoge las normas y restricciones con las que se ha de visualizar el documento HTML. Esto, en cambio, se simplifica mucho en HTML5 ya que no tiene DTD asociado.

Navegadores que soportan esta declaración

				
Chrome	Firefox	IE Explorer	Ópera	Safari

Declaraciones DOCTYPE según el tipo de documento

- HTML 4.01 Strict
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`
- HTML 4.01 Transitional
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`
- HTML 4.01 Frameset
`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">`
- XHTML 1.0 Strict

- ```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```
- XHTML 1.0 Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```
- XHTML 1.0 Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```
- XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```
- HTML5

```
<!DOCTYPE html>
```

Según la declaración anterior, la estructura básica de una página HTML5 en español es:

```
<!DOCTYPE html>
<html lang="es">
 <head>
 <title> Título de la página</title>
 Elementos de cabecera
 .
 .
 </head>
 <body>
 Elementos y contenido de la página
 .
 .
 </body>
</html>
```

Ejercicio nº.1: Una página html5 es un archivo texto como extensión .htm ó .html con la estructura que hemos visto anteriormente y compuesto de etiquetas y texto. Para crear una página web básica, solo nos hace falta un simple editor de textos y escribir tanto etiquetas como texto. Vamos a crear una página web con notepad++ llamada primera.html

que visualice por pantalla el mensaje “Mi primer documento html” y tenga como título “Mi primer documento html”. Después de creada, ábrela con tu navegador favorito y observa el resultado

#### Vídeo primeranotepad.mp4

Fíjate ahora como se haría una página simple con el editor de web bluegriffon

#### Vídeo primeragriffon.mp4

### 2.4.2. Cabecera de un documento html. <head>

Las páginas y documentos htl incluyen más información de la que los usuarios ven en sus pantallas. Estos datos adicionales siempre están relacionados con la propia página, por lo que se denominan *metainformación* o *metadatos*. La metainformación siempre se incluye en la sección de la cabecera, es decir, dentro del elemento <head>

Aunque la metainformación más conocida y utilizada es el título de la propia página, se puede incluir mucha otra información útil para los navegadores y para los buscadores.

	En <head> no se puede incluir contenido de la página web ya que no se visualiza con el navegador
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------

Etiqueta	<head>					
Descripción	Define la cabecera de un archivo html					
Elemento	No es bloque ni inline					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales					
Atributos propios						
Diferencia entre html	El atributo profile no lo soporta HTML5					

## 4.01 y html5

Los elementos más importantes de <HEAD> son:

Elementos <head>	Descripción
<title>     	Define el título de un documento
<base>     	Define una dirección por defecto o una página por defecto de todos los enlaces de una página
<link>     	Define una relación entre un documento y un recurso externo
<meta>     	Información adicional de un documento html.
<script>     	Define un script javascript local
<style>     	Define los estilos de un documento html

	<i>Todos estos elementos soportan los atributos globales HTML.</i>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------

- **<title>** Especificará el título del documento html. Dentro de esta etiqueta no se podrá usar ninguna de las restantes etiquetas html.

	<i>El título no forma parte del contenido del documento HTML y es obligatorio incluirlo.</i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------

**Ejemplo:**

<head>  <title>Titulo de la página web</title>
------------------------------------------------------

```
</head>
```

El elemento <title>:

- Se muestra en la barra de herramientas (toolbar) del navegador.
- Proporciona el título de la página cuando se añade a los favoritos del navegador.
- Muestra el título de la página en los resultados de los motores de búsqueda.

- **<base>** Especifica la dirección o marco destino para todas las direcciones relativas que se encuentren en la página.

La razón más importante para usar <base> es asegurar que cualquier URL relativa dentro del documento se resolverá en una dirección correcta, aun si el documento se mueve o renombra.

Atributos propios	Valor	Descripción
href 	URL	Especifica la dirección base para todas las URL relativas del documento.
target 	_blank _parent _self _top Nombre marco	Especifica el lugar por defecto donde se visualizará un archivo o página enlazada.

### Ejemplo:

```
<head>
 <base href="http://www.mentor.mec.es">
</head>
```

- **<link>** Define una relación entre un documento y otro. Su uso más importante es para la referencia a archivos de estilos.

Atributos propios	Valor	Descripción
href 	URL	Especifica la localización del archivo al que se hace referencia
hreflang	Código idioma	Especifica el idioma del documento

		enlazado
media 	Dispositivo	Especifica el dispositivo para el cual está optimizado el documento
rel 	alternate archives author bookmark external first help icon last license next nofollow noreferrer pingback prefetch prev search sidebar stylesheet tag up	Especifica la relación entre el propio documento y el enlazado.
sizes 	Anchoxalto	Especifica el tamaño en pixels de un icono. Solo se puede usar con rel="icon". Por ahora no lo soporta ningún navegador.
type 	Tipo MIME	Especifica el tipo MIME del documento al que se hace referencia

 → Nuevo en HTML5

### Ejemplo:

```
<head>

<link href="estilos.css" rel="stylesheet" type="text/css">

</head>
```

En este ejemplo se hace referencia al archivo estilos.css que una hoja estilos (rel="stylesheet") y que se reconoce con el tipo MIME "text/css".

- **<meta>** Da información adicional sobre la página que puede ser empleada por los navegadores y por los buscadores.

Atributos propios	Valor	Descripción
charset 	Mapa de caracteres	Especifica el mapa de caracteres de un documento html
content 	Texto	Especifica el valor que toma el atributo <i>name</i> o <i>http-equiv</i>
http-equiv 	content-type default-style refresh	Especifica un cabecera HTTP de información o valor del atributo <i>content</i>
name 	application-name author description generator keywords	Especifica el autor, nombre de aplicación, descripción, con qué paquete ha sido generado y palabras clave de un documento html.

 → Nuevo en HTML5

### Ejemplo:

```
<head>

 <meta name="author" content="María del Mar">

 <meta name="description" content="Página de astronomía">

 <meta name="keywords" content="estrellas, cielo, constelaciones, Luna">

</head>
```

En este se indica que la autora de esta página es María del mar, que trata de astronomía y que los buscadores de información de Internet van a asociar dicha página con los términos "estrellas", "cielo", "constelaciones" y "Luna" siempre y cuando esta página esté alojada en un servidor web de Internet

	<i>charset en HTML5 reemplaza a http-equiv a la hora definir el mapa de caracteres de una página web.</i>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------

	<i>UTF-8 es el conjunto de caracteres más empleado en Internet</i>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------

#### Ejemplo HTML5:

```
<head>

 <meta charset="utf-8">

</head>
```

#### Ejemplo HTML 4.01:

```
<head>

<meta http-equiv="content-type" content="text/html; charset=UTF-8">

</head>
```

- **<script>** Es usado para incluir un script o programa local javascript en el documento html.

Atributos propios	Valor	Descripción
async      	async	Especifica que el script se ejecutará de forma asíncrona (solo scripts externos)
charset      	Mapa de caracteres	Especifica el mapa de caracteres usado en un script externo
defer      	defer	Especifica que el script se ejecuta al final del análisis del archivo. No soportado en el navegador Opera 12 ni anteriores versiones.
src      	URL	Especifica la localización del script al que se hace referencia-
type      	Tipo MIME	Especifica el tipo MIME del script al que se hace referencia

 → Nuevo en HTML5



**Ejemplo:**

```
<head>

 <script>

 document.getElementById("elemento").innerHTML = "Hola";

 </script>

</head>
```

En este ejemplo se escribe "Hola" dentro del elemento html con id="elemento"

- **<noscript>**: Cuando se intenta ejecutar un script define un contenido alternativo para los navegadores que tengan deshabilitada esta opción o no la soporten.

	<i>&lt;noscript&gt; en HTML5 puede ir tanto en &lt;body&gt; como en &lt;head&gt; a diferencia de HTML4 que solo puede ir en &lt;body&gt;</i>
-----------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------

**Ejemplo:**

```
<script>

 document.getElementById("elemento").innerHTML = "Hola";

</script>

<noscript> Tu navegador no soporta JavaScript</noscript>
```

**2.4.2. 1. HTML5 y los caracteres especiales castellanos (ñ, á, ü, etc.).**

Los caracteres o, más genéricamente, símbolos que nosotros escribimos en un documento no se pueden representar como tales dentro de las aplicaciones por lo que necesitamos algún tipo de conversión que permita transformar esos caracteres en símbolos que las aplicaciones puedan entender y manejar. Esta equivalencia se refleja en lo que denominamos **tabla de caracteres**

Para visualizar correctamente una página web, el navegador debe conocer la tabla de caracteres en la que está codificada dicha página y así realizar la conversión correcta.

En las primeras versiones de HTML se usa el código ASCII como tabla de caracteres por defecto, es decir, que si no se especificaba lo contrario el documento se interpretaba que estaba codificado en ASCII.

Más adelante Microsoft incorporó el código ANSI (Windows-1252) a sus sistemas operativos (en realidad ANSI es una extensión de ASCII que pretendía ampliarlo) y muchas aplicaciones lo tomaron como tabla de caracteres por defecto.

Con la aparición de HTML4.01, las aplicaciones utilizaban como tabla de caracteres por defecto ISO-8859-1 que también era una extensión de ASCII con muchas similitudes con ANSI aunque no tan amplia (muchos navegadores en sus versiones más antiguas interpretaban las páginas escritas en ISO-8859-1 como ANSI debido a esa similitud).

Pero tanto las anteriores codificaciones no llegaban a representar todos los caracteres internacionales que se pueden utilizar por lo que surgió el código UNICODE con la finalidad de paliar la carencia anterior. De aquí surgió la tabla de caracteres UTF-8 que es la que por defecto se utiliza en aplicaciones para HTML5.

Después de esta introducción debemos saber que la codificación de las páginas web (charset) depende de si:

- El editor en que se ha escrito el documento HTML usa por defecto UTF-8 o ISO-8859-1. Si el archivo original estaba escrito en ISO-8859-1 y se edita en UTF-8, los caracteres específicos castellanos se verán mal codificados
- La configuración del servidor web en el caso de que esté alojada en su estructura la página web.
- Si se especifica el conjunto de caracteres en la etiqueta META, atributo charset, de la sección HEAD de la página web.

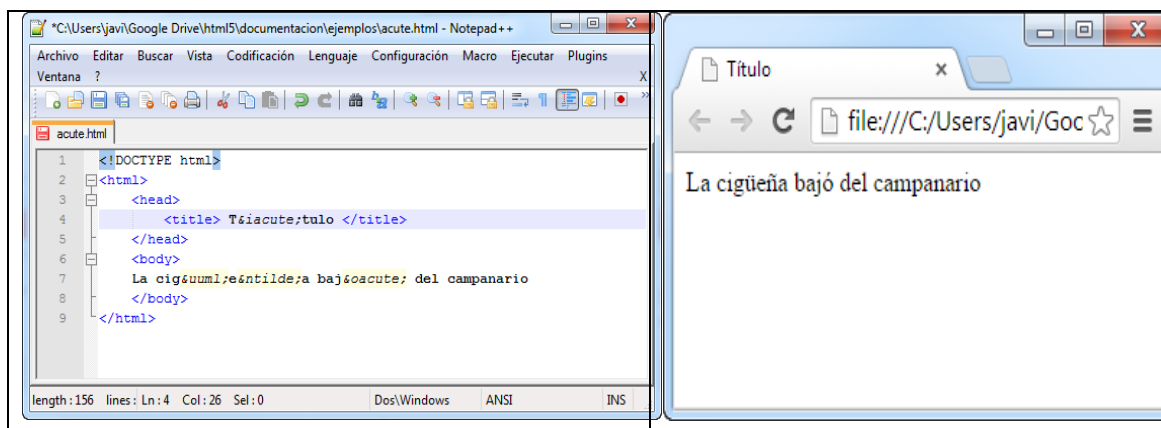
No obstante y a modo de ejemplo, el estándar HTML representa de la siguiente manera algunos de los caracteres especiales del lenguaje castellano

:

Carácter	Representación HTML	Carácter	Representación HTML
Á	&Aacute;	á	&aacute;
É	&Eacute;	é	&eacute;
Í	&Iacute;	í	&iacute;
Ó	&Oacute;	ó	&oacute;
Ú	&Uacute;	ú	&uacute;
À	&Agrave;	à	&agrave;
È	&Egrave;	è	&egrave;
Ö	&Ouml;	ö	&ouml;
Ü	&Uuml;	ü	&uuml;
Ñ	&Ntilde;	Ñ	&ntilde;
<	&lt;	>	&gt;
&	&amp;	“	&quot;

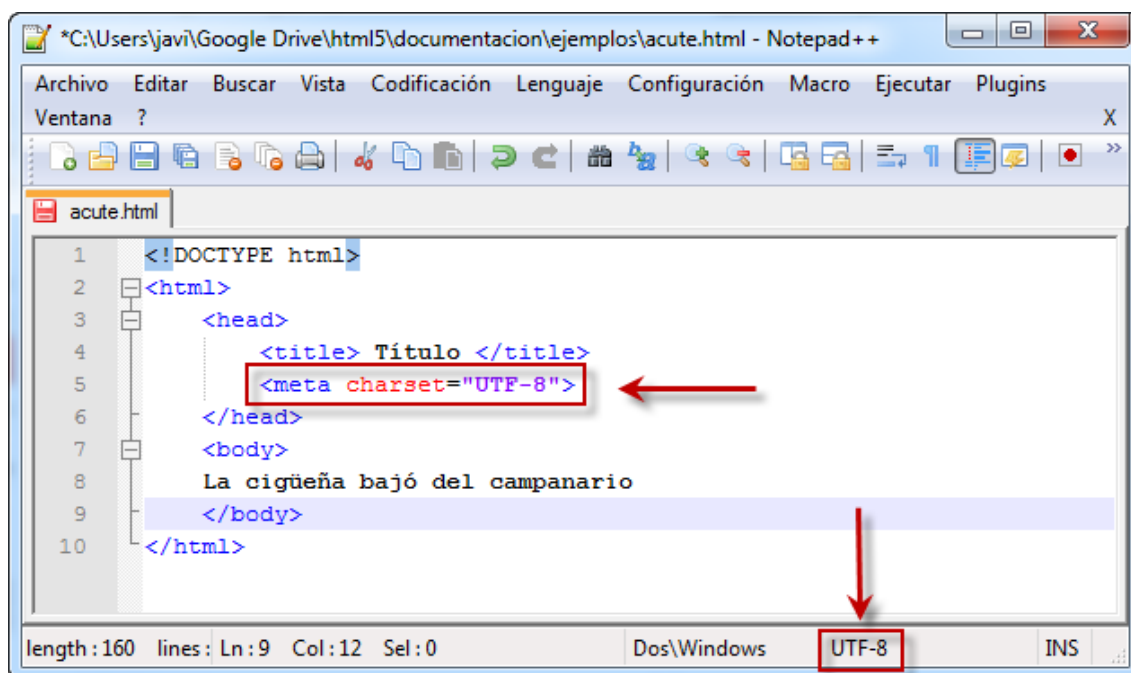
Utilizando las representaciones anteriores nunca habrá ningún problema a la hora de visualizar esos caracteres en un navegador independientemente de la codificación que se haya usado para hacer el documento.

Ejemplo:



No obstante, el mapa de caracteres UTF-8 evita los problemas a la hora de representar los caracteres especiales castellanos aunque puede haberlos con algún carácter de lenguajes orientales.

En resumen, lo más aconsejable es crear nuestra página web con un editor que la codifique en UTF-8 y agregar la etiqueta `<meta charset="UTF-8">` a la sección `<head>` de dicha página.



### 2.4.3. Cuerpo de un documento html. `<body>`

El elemento `<BODY>` contiene tanto el texto como otros elementos o etiquetas que hacen que la información de una página web sea visible con un determinado aspecto a través de un navegador.

Etiqueta	<code>&lt;body&gt;</code>
Descripción	Define el cuerpo de un documento html

Elemento	Bloque				
Navegadores que la soportan					
		Chrome	Firefox	IE Explorer	Ópera
Atributos	Globales y eventos				
Atributos propios					
Diferencia entre html 4.01 y html5	En html5 se suprimen todos los atributos que modifican el aspecto general de un documento HTML tales como background, bgcolor, etc.				

Tanto en html4 como en xhtml y otros lenguajes de marcas más antiguos el elemento <BODY> permitía, a través de sus atributos, dar aspectos generales tanto al entorno como al contenido de un documento html como, por ejemplo, dar un color de fondo a la página (<BODY bgcolor="#red">).

En html5 se usan hojas de estilo CSS para aplicar cualquier modificación de aspecto de <BODY> y los atributos que tenía esta etiqueta en html4 o xhtml desaparecen como tales.

Si tienes dudas de cómo se pueden utilizar los caracteres especiales castellanos mira estos dos vídeos (uno para notepad++ y otro para BlueGriffon)

Utfnotepad.mp4

Utfgriffon.mp4

**Ejercicio nº. 2:** Seguimos utilizando notepad++ para crear una página html que respete la estructura básica de una página html5 y contenga lo siguiente con el fin de practicar con las partes <head> y <body>:

- Título: Página de lengua castellana
- Autor: El creador de la misma
- Descripción: Una descripción adecuada a una página de castellano
- Palabras clave: Palabras relacionadas con la lengua castellana
- Código de caracteres: El que no nos de problemas a la hora de poner ñ o acentos.
- Como cuerpo de la página, o sea, contenido de la misma, tendrá solamente este texto:

Probando códigos \ ñ í Ú @ & á Ñ Ç ç

Guárdala como segunda.html y visualízala con Chrome y algún otro más navegador para ver sus diferencias y si los caracteres se reproducen fidedignamente.

## 2.5. VALIDACIÓN DE UNA PÁGINA WEB

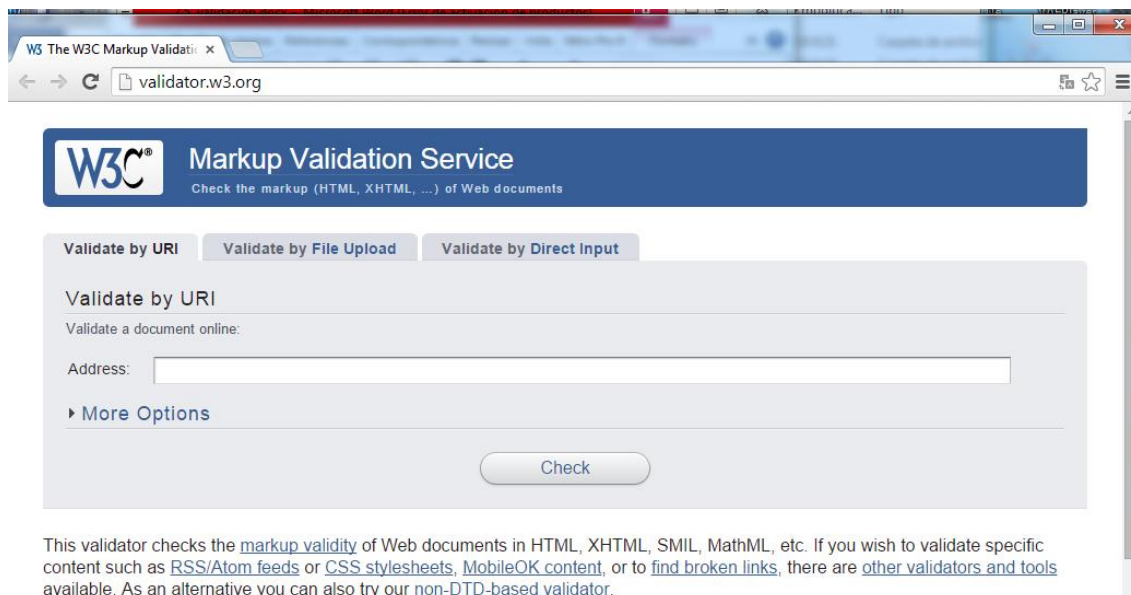
El proceso de validación es aquel que indica si un documento escrito en un determinado lenguaje cumple las especificaciones del mismo, es decir, si respeta sus normas.

En el ámbito del diseño web la validación de una página no es obligatoria ya que un documento html puede ser reproducido correctamente por un navegador aunque no cumpla las normas del lenguaje en el que está escrito aunque siempre es mejor que lo haga.

En versiones anteriores de HTML a HTML5 y en XML y sus derivados (XHTML) estas especificaciones vienen reflejadas en el DTD o Document Type Definition (Definición del Tipo de Documento). HTML5 no tiene DTD al no basarse en SGML.

	<i>El organismo W3C se encarga de mantener los DTD tanto de los lenguajes HTML como XML</i>
-----------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------

El organismo W3C ofrece una herramienta pública a través de la cual se puede saber si una página web cumple las normas del lenguaje en la que está escrita y que es accesible a través de la dirección <http://validator.w3.org>



Esta página nos permite validar páginas de tres maneras diferentes:

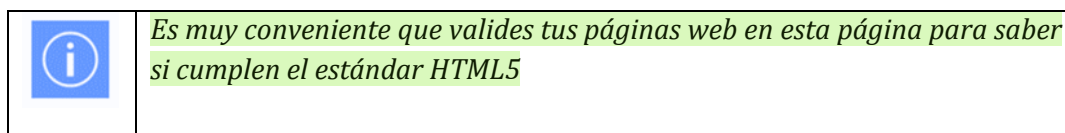
- **Validate by URI:** Esta opción solo sirve para páginas que están alojadas en algún servidor de Internet como, por ejemplo, <http://www.mentor.mec.es> por lo que se escribe la dirección de Internet para validar dichas páginas.

- **Validate by File Upload:** Esta opción permite subir un documento html desde nuestro equipo y validarlo.
- **Validate by Direct Input:** Esta opción permite validar partes o páginas enteras escritas directamente en un editor. Esta opción es muy útil cuando se quiere saber determinados trozos de código son válidos para insertarlos en una página web.

Si el documento html llevado por cualquiera de los métodos anteriores al proceso de validación no lo supera se mostrará un informe completo de los errores que tuviera.

En este vídeo verás cómo se valida una página web

#### validación.mp4



**Ejercicio nº. 3:** A través de la opción **Validate by File Upload** de la página <http://validator.w3.org>, valida la página del ejercicio nº. 2 del tema anterior comprobando si es un documento HTML5 y, en el caso de que hubiera errores, intenta corregirlos (recuerda que los warnings no son errores).

**Ejercicio nº. 4:** Valida cualquiera de las páginas web que sueles visitar (puedes usar <http://google.es>) indicando:

- Dirección de la página.
- Mapa de caracteres
- Tipo de documento
- Tres errores, si los tuviera, intentando dar una solución a cada uno de ellos.

## 2.6. COMENTARIOS

Un comentario es texto que se inserta en una página web para que, cuando se accede al código html de dicha página, explique una parte muy concreta del documento HTML.

	<i>Los comentarios no aparecen cuando se visualiza la página web con un navegador pero sirven de información sobre algún elemento o parte del documento.</i>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------

	<i>Agrandan el documento HTML con información que no se va a ver</i>
-----------------------------------------------------------------------------------	----------------------------------------------------------------------

Para insertar un comentario primero se abre con `<!--` , se pone el texto del comentario y se cierra con `-->`

### Ejemplo:

```
<!-- Esto es un comentario -->
```

### Ejemplo completo:

```
<!DOCTYPE html>
<html lang="es-es">
 <head>
 <meta charset="utf-8">
 <!-- Debajo va el título de la página -->
 <title>Título</title>
 </head>
 <body>
 <!-- Uso la etiqueta de párrafo -->
 <p>
 Esto es un párrafo
 </p>
```

```
 <!-- Aquí acaba la página -- >

 </body>

</html>
```

En el navegador devolvería lo siguiente:

Esto es un párrafo

Es decir, la información que va entre <p> y </p>.



## 2.7. DIVISIONES, PÁRRAFOS Y AGRUPACIONES

A diferencia de la mayoría de los procesadores de texto, HTML utiliza etiquetas de división explícita (<div>), párrafo (<p>), salto de línea (<br>) y agrupación de varios elementos dentro de un bloque (<span>) para controlar la alineación y flujo del texto.

- <br>: Devuelve un retorno de carro o salto de línea. No tiene etiqueta de cierre, es decir, no existe </br> ya es una etiqueta vacía

Etiqueta	 					
Descripción	Devuelve un retorno de carro					
Elemento	En línea y etiqueta vacía					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos					
Atributos propios	No tiene					
Diferencia entre html 4.01 y html5	No hay					

- <div>: Divide el documento en secciones distintas e independientes y se puede utilizar estrictamente como una herramienta organizativa, sin ningún tipo de formato asociado.

	<i>Su aspecto se maneja con estilos CSS</i>
-------------------------------------------------------------------------------------	---------------------------------------------

Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <meta charset="utf-8">
```

```
<title>Div y span</title>

</head>

<body>

 <div style="border: 1px solid red;">

 Cabecera del documento html

 </div>

 Texto entre la cabecera y el cuerpo

 <div style="border: 1px solid green;">

 Cuerpo del documento html

 </div>

 Texto entre el cuerpo y el pie

 <div style="border: 1px solid blue;">

 Pie del documento html

 </div>

</body>

</html>
```

Cabecera del documento html

Texto entre capas

Cuerpo del documento html

Texto entre capas

Pie del documento html

En este ejemplo se crean tres divisiones en un documento html (una para la cabecera, otra para el cuerpo del documento y otra para el pie del mismo) y entre cada división hay un texto.

Para el aspecto de la división se emplean estilos que permiten crear un borde de color de un determinado grosor.

Etiqueta	<div>				
Descripción	Define una división en una página html				
Elemento	En bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos				
Atributos propios	No tiene				
Diferencia entre html 4.01 y html5	HTML5 no soporta el atributo align				

- <p>: Señala el inicio de un párrafo que finaliza con </p>

	<i>Su aspecto se maneja con estilos CSS</i>
-------------------------------------------------------------------------------------	---------------------------------------------

Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <meta charset="UTF-8">

 <title>Divisiones y párrafos</title></head>

 <body>

 <p style="color:yellow;background-color:grey ;">
```

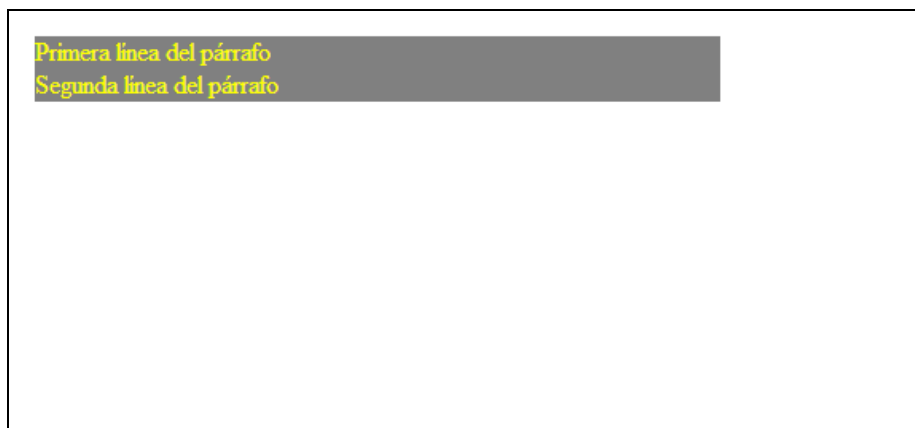
```

Primera línea del párrafo

Segunda línea del párrafo

</p>
</body>
</html>

```



En este ejemplo el párrafo definido tiene un color de fondo gris y un color de texto amarillo.

Etiqueta	<p>					
Descripción	Define un párrafo					
Elemento	bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y de eventos					
Atributos propios	No tiene					
Diferencia entre html 4.01 y html5	HTML5 no soporta el atributo align					

- <span>: Permite agrupar varios elementos en línea seguidos dentro de un mismo bloque para después darles formato con hojas de estilo CSS.

Etiqueta	<span>					
Descripción	Define elementos en línea dentro de un bloque					
Elemento	En línea					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos					
Atributos propios	Ninguno					
Diferencia entre html 4.01 y html5	Ninguna					

<span> y <div> tienen las mismas propiedades, pero la diferencia varía en la función que cumplen, ya que mientras la etiqueta span trabaja sólo como contenedor en línea, la etiqueta div trabaja como contenedor de bloque.

Cabecera del documento html

Texto entre la cabecera y el cuerpo

Dentro del cuerpo del documento html vamos a crear dos partes en línea la primera parte tiene color rojo, texto que no pertenece a ningún contenedor inline y la tercera parte un color azul y un tamaño de 30 pixels

Texto entre el cuerpo y el pie

Pie del documento html

	<i>En HTML5 se usan las llamadas etiquetas semánticas que suplen las funciones de div y span</i>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------

### 2.7.1. Ejercicios

Para los ejercicios siguientes puedes usar el editor web BlueGriffon o Notepad++, pon un título coherente en cada página (recuerda que el título va en la sección <HEAD> por lo que no tienes que escribirlo como contenido en <BODY>) y después de guardar cada ejercicio ábrelo con, al menos, tres de tus navegadores para ver las diferencias que pueda haber al visualizarlos con un navegador u otro.

**Ejercicio nº. 5:** Como aperitivo a la estructura de una página html5, vamos a dividir nuestra página web en partes significativas. Fíjate en la imagen y con bluegriffon o notepad++ crea las divisiones con el elemento <div>. Aunque no hemos visto nada aún de estilos los vamos a utilizar para ver mejor el resultado. Aquí tienes unas pistas para hacer el ejercicio:

- Antes de empezar y por si no conoces cómo funcionan los colores en HTML pásate por el apéndice de colores (enlace al apéndice de colores)
- Pon el título “Divisiones” a la página, usa español (España) para el lenguaje de los elementos de la página y mapa de caracteres UTF-8
- Encabezado: <div style="background-color: grey; height: 80px;">  
Lo que viene a decir que el color de fondo (background-color) es gris y el ancho (height) de <div> es 80 pixels.
- Cuerpo de la página: <div style="background-color: #ccffff; height: 200px;">
- Pie de página: <div style="height: 80px; background-color: #ffcc33;">
- No tengas en cuenta el tamaño de la letra ya que simplemente está puesto así para que se vea mejor. Escribe el texto sin negrita, de tamaño normal y puede estar en cualquier parte de la división, es decir, puede estar pegado, por ejemplo, al borde superior de la misma.
- Guárdalo como ejercicio3.html

**Vídeo: divisiones.mp4**

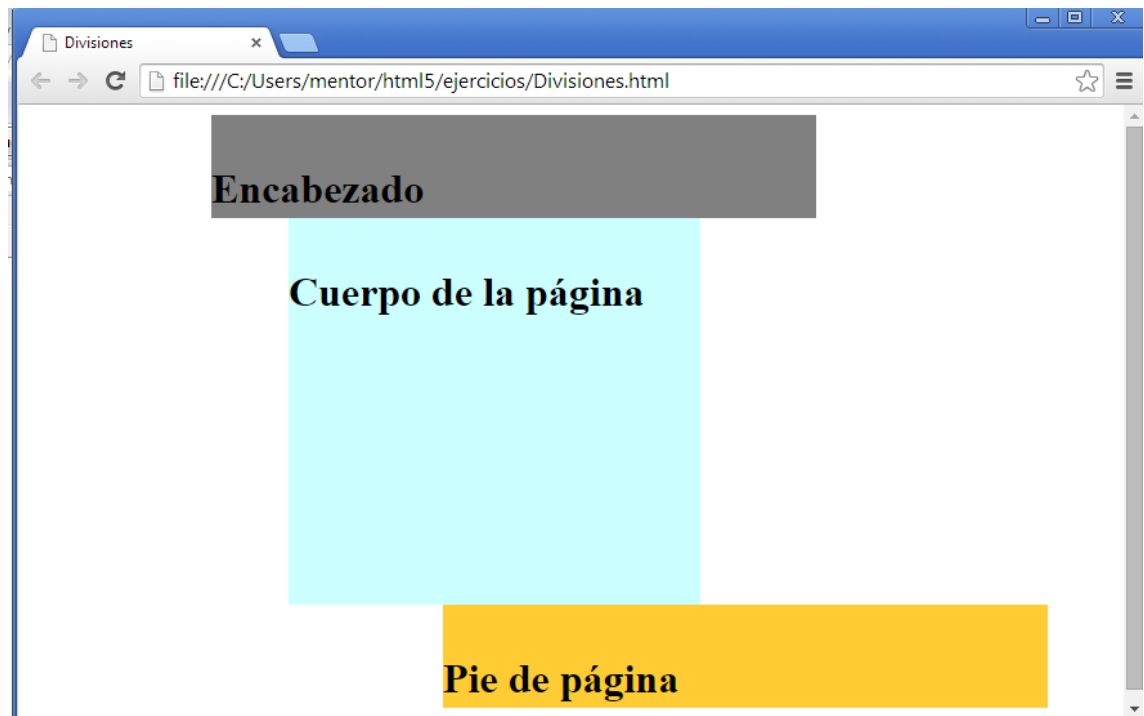
## Encabezado

## Cuerpo de la página

## Pie de la página

**Ejercicio nº. 6:** Vamos a trabajar con los anchura de las divisiones anteriores y su relación con el cuerpo del documento (<BODY>))

- Abre el ejercicio anterior con BlueGriffon o notepad++.
- Cambia el título por “Anchura de divisiones”
- Para los puntos siguientes vamos a usar otras opciones de estilos como son:
  - o *margin-left: 150px;* que significa que el margen izquierdo está a 150 pixels del borde de su contenedor (en el caso de que fuera <BODY> lo sería del borde izquierdo de la ventana)
  - o *width: 470px;* que indica que el elemento sobre el que se aplica tiene 470 pixels de ancho.
- Modifica la anchura de <BODY> dejando el su borde izquierdo a 150 pixels del borde la página y que tenga una anchura de 650 pixels. Observa como las divisiones que has creado se adaptan al ancho de <BODY> debido a que <BODY> es un elemento de tipo bloque lo que implica que el ancho de los elementos que hay dentro se ajustan al suyo.
- Modifica el ancho del encabezado y déjalo con 470 pixels de ancho.
- Modifica el cuerpo de la página dejando un margen a la izquierda de 60 pixels y con un ancho de 320 pixels
- Modifica el ancho del pie de página dejando un margen a la izquierda de 180 pixels.
- Guarda la página como ejercicio4.html
- Fíjate en la imagen ya que es lo que tienes que ver en el navegador cuando abras con él la página (no tengas en cuenta el tamaño y posición del texto).



**Vídeo: anchodivisiones.mp4**

**Ejercicio nº. 7:** Vamos a practicar con el elemento `<span>`. Este elemento es in-line y sirve para hacer subdivisiones normalmente dentro de una división. Abre el ejercicio “divisiones.html” con BlueGriffon o notepad++ y vamos a crear lo siguiente dentro de “Cuerpo de la página” y debajo del texto que pusimos:

- Cambia el título de la página por “Span”
- Un elemento `<span>` con el texto “Primera subdivisión” con color de fondo #33cc00
- El texto “Estoy en medio de dos `<span>`”
- Otro elemento `<span>` con el texto “Segunda subdivisión” de color rojo (red). Usa el estilo “color” para el texto de este elemento y aplícalo sobre `<span>` no sobre dicho texto.
- No tengas en cuenta el tamaño del texto ni su posición pero si el color y la disposición del mismo con arreglo a la imagen de abajo.
- Guarda el archivo como ejercicio5.html



## Encabezado

## Cuerpo de la página

Primera subdivisión Texto en medio Segunda subdivisión

## Pie de la página

**Ejercicio nº. 8:** Vamos con los párrafos y seguimos avanzando un poco en el tema de aplicar estilos a los diferentes elementos.

- Abre el ejercicio3.html.
- Cambia el título de la página por “Párrafos”
- En la primera división, “encabezado”, borra el texto que hay y crea un párrafo con el color de fondo #ff99ff y el texto “Párrafo de encabezado”.
- Ajusta el ancho de la segunda división o “cuerpo de la página” a 150 pixels
- En la división cuerpo de la página, borra el texto que hay y haz lo siguiente:
  - o Inserta un retorno de carro (<br>)
  - o Crea un párrafo con el texto “Estoy en el segundo párrafo del cuerpo de página”, con el fondo de color azul, el texto de color #ffff99 y centrado.
  - o Crea otro párrafo con el texto “Tercer párrafo”, con el fondo de color #ffccff y ajustado el texto a la derecha.
- Guarda la página como ejercicio6.html.
- Fíjate en cómo se ajustan los párrafos a la anchura de las divisiones y que los dos párrafos de la segunda división o “cuerpo de la página” están uno debajo del otro debido a que <p> es un elemento de tipo bloque.

Párrafo de encabezado

Estoy en el segundo  
párrafo del cuerpo de  
página

Tercer párrafo

Pie de la página

**Ejercicio nº.9:** Ahora vamos a ver cómo podemos poner dos párrafos uno al lado del otro, es decir, como cambiar un elemento de tipo bloque para que funcione como en línea o inline.

- Abre el archivo ejercicio3.html.
- Borra el texto de la división “Cuerpo de la página” y haz lo siguiente en esa división:
  - o Déjale un ancho de 700px.
  - o Inserta un retorno de carro (<br>)
  - o Crea un párrafo que tenga de ancho 550px, un color de fondo #009999, un color de texto de #33cc00 y el siguiente texto “Y este es el primer párrafo del cuerpo de página del ejercicio séptimo del bloque de divisiones del tema GENERALIDADES HTML”.
  - o Crea otro párrafo que tenga de ancho 335px, un color de fondo blanco, un color de texto negro y el siguiente texto “Ahora estoy en el segundo párrafo del cuerpo de página”
  - o Y, por último, crea otro párrafo que ocupe 225px, un color de fondo de #99fff9, un color de texto de #6666cc
  - o La primera imagen es la que te muestra cómo te deben quedar los párrafos en primera instancia.
  - o Ahora cambia las propiedades de los párrafos para que queden uno junto a otro tal y como te muestra la segunda imagen.
  - o Fíjate que ahora los párrafos ya no cumplen el ancho que les has puesto en los puntos anteriores debido a que ahora son elementos de tipo inline.
  - o Guarda la página como ejercicio7.html.

Encabezado

Y este es el primer párrafo del cuerpo de la página del ejercicio séptimo del bloque de divisiones del tema GENERALIDADES HTML

Ahora estoy en el segundo párrafo del cuerpo de la página del ejercicio

Estoy en el tercer párrafo del cuerpo de página

Pie de la página

Encabezado

Y este es el primer párrafo del cuerpo de la página del ejercicio séptimo del bloque de divisiones del tema GENERALIDADES HTML Ahora estoy en el segundo párrafo del cuerpo de la página del ejercicio Estoy en el tercer párrafo del cuerpo de página

Pie de la página

## 2.8. TABLAS

Una tabla es una colección de filas donde cada una de ellas contiene un número determinado de celdas. Dentro de estas celdas se puede introducir texto, imágenes o cualquier otro elemento html incluidas otras tablas.

	<i>Las tablas en HTML5 solo valen para presentar datos tabulados no para organizar los elementos de una página como se usaba con HTML4</i>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------

- **<table>** Define, empleando normalmente estilos CSS. los parámetros generales de la tabla que marcan su aspecto.

El contenido de la tabla está entre la etiqueta de apertura <table> y la de cierre </table>

Etiqueta	<table>				
Descripción	Define una tabla de datos				
Elemento	Bloque				
Navegadores que la soportan	    				
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios	sortable				
Diferencia entre html 4.01 y html5	Los atributos de aspecto como align, bgcolor, border, etc. están suprimidos en HTML5				

Atributos propios	Valor	Descripción
sortable	sortable	Especifica que la tabla debería estar ordenada

Dentro de <table> </table> se definen otros elementos que marcan el aspecto de la tabla.

- **<th>**: Etiqueta que indica que la celda es de cabecera o título de una columna, fila, conjunto de filas o conjunto de columnas. El texto del título se pone en negrita y centrado por defecto.

Etiqueta	<th>					
Descripción	Define la cabecera de fila, grupo de filas, columna, grupo de columnas de una tabla					
Elemento	Bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					
Atributos propios	scope					
Diferencia entre html 4.01 y html5	Los atributos de aspecto como align, bgcolor,, etc. están suprimidos en HTML5					

Atributos propios	Valor	Descripción
abbr 	texto	Descripción corta del contenido de la celda de cabecera
colspan 	número	Especifica el número de columnas que se pueden fusionar en una celda de cabecera. Solo Firefox permite colspan="0"
headers 	identificador	Indica un identificador que relaciona celdas de cabecera entre si
rowspan 	número	Especifica el número de filas que se pueden fusionar en una celda de cabecera
scope 	col colgroup row rowgroup	Indica si es cabecera de una columna, fila, conjunto de filas o conjunto de columnas
sorted	Reversed	Define el sentido del orden de una columna

	Number	
	Reversed number	
	Number reversed	

- `<tr>`: Etiqueta que define una fila. La fila finaliza con el etiqueta `</tr>`.

Etiqueta	<code>&lt;tr&gt;</code>					
Descripción	Define una fila de una tabla					
Elemento	Bloque					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos					
Atributos propios						
Diferencia entre html 4.01 y html5	Los atributos de aspecto como align, bgcolor,, etc. están suprimidos en html5					

- `<td>`: Etiqueta que define una celda. Su texto se pone normal y ya alineado a la izquierda por defecto. La celda finaliza con la etiqueta `</td>`

Etiqueta	<code>&lt;td&gt;</code>					
Descripción	Define una celda en una tabla					
Elemento	Bloque					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari

Atributos	Globales y eventos
Atributos propios	
Diferencia entre html 4.01 y html5	Los atributos de aspecto como align, bgcolor,, etc. están suprimidos en html5



*Una tabla tiene el mismo número de celdas en todas las filas aunque se pueden agrupar.*

Ejemplo de tabla simple:

```
<!DOCTYPE html>
<html lang="es-es">
 <head>
 <title>Col y colgroup</title>
 <meta charset="UTF-8">
 </head>
 <body>

 <table>
 <tr>
 <th> Titulo columna 1</th>
 <th> Título columna 2</th>
 <th> Título columna 3</th>
 </tr>
 <tr>
 <td> fila 1 celda 1 </td>
 <td> fila 1 celda 2 </td>
```

```

 <td> fila1 celda 3 </td>

 </tr>

 <tr>

 <td> fila 2 celda 1 </td>

 <td> fila 2 celda 2 </td>

 <td> fila 2 celda 3 </td>

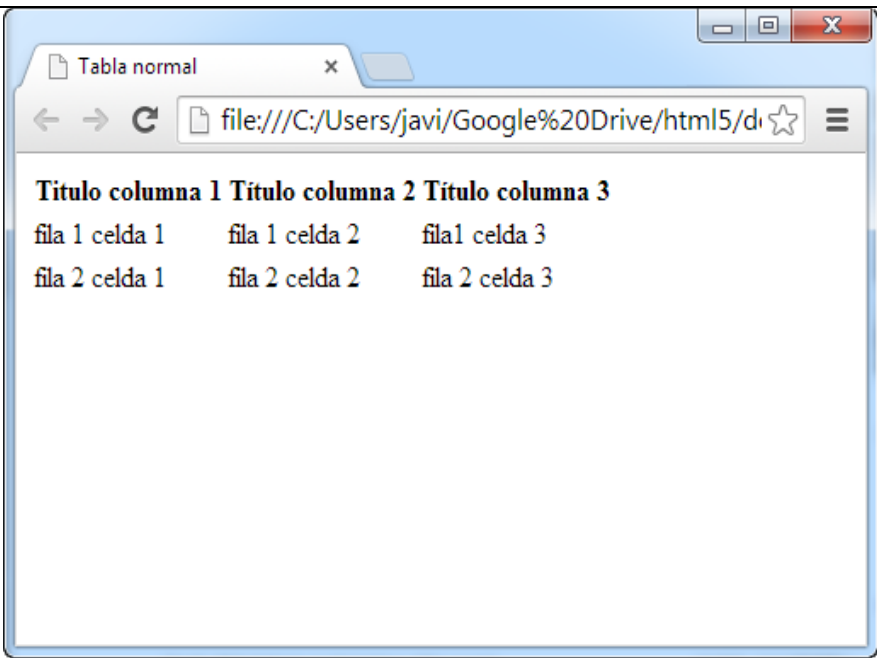
 </tr>

</table>

</body>

</html>

```



Titulo columna 1	Titulo columna 2	Titulo columna 3
fila 1 celda 1	fila 1 celda 2	fila1 celda 3
fila 2 celda 1	fila 2 celda 2	fila 2 celda 3

En este ejemplo la tabla sale sin borde por lo que no se distingue la separación entre las celdas. Hay que evitar atributos como border (que dibuja los bordes de las celdas) que están totalmente obsoletos y usar hojas de estilos para hacer este tipo de cosas.

- <caption> Título de una tabla que debería dar una descripción breve del contenido de la tabla. Tiene que ir justo debajo de la etiqueta de apertura de la tabla, es decir, debajo de <table>
-



Etiqueta	<caption>					
Descripción	Define el título de una tabla					
Elemento	En línea					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					
Atributos propios						
Diferencia entre html 4.01 y html5	El atributo align está suprimido en HTML5					

	<i>Solo puede haber una etiqueta &lt;caption&gt; por tabla</i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------

- <colgroup> Define un grupo de columnas de una tabla para darles un formato común, es decir, se pueden aplicar el mismo estilo a un conjunto de columnas. Va detrás justo de <caption> y delante de <thead>

Etiqueta	<colgroup>					
Descripción	Especifica un grupo de una o más columnas en una tabla					
Elemento	Bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					
Atributos propios	span					
Diferencia entre html 4.01 y html5	Los atributos de aspecto como align, bgcolor,, etc. están suprimidos en html5					

Atributos propios	Valor	Descripción
Span 	número	Especifica el número de columnas contiguas que se pueden agrupar

- `<col>` Especifica una columna en un tabla y permite dar estilos a una columna en concreto. Normalmente es componente de `<colgroup>` aunque se puede utilizar como etiqueta independiente.

Etiqueta	<code>&lt;col&gt;</code>					
Descripción	Especifica una columna en un tabla					
Elemento	Bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					
Atributos propios	span					
Diferencia entre html 4.01 y html5	Los atributos de aspecto como align, bgcolor,, etc. están suprimidos en HTML5					

Atributos propios	Valor	Descripción
Span 	número	Especifica el número de columnas contiguas que se pueden agrupar

	<i>El atributo span tiene el mismo uso tanto para <code>&lt;colgroup&gt;</code> como para <code>&lt;col&gt;</code>. Esto es debido a que <code>&lt;col&gt;</code> puede ser usado como elemento independiente, es decir, que no esté dentro de <code>&lt;colgroup&gt;</code></i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Ejemplo:**

```
<!DOCTYPE html>

<html lang="es-es">

<head>

<title>Col y colgroup</title>

<meta charset="UTF-8">

</head>

<body>

<table>

 <colgroup>

 <col style="background-color:orange">

 <col span="2" style="background-color:yellow">

 <col style="background-color:grey">

 </colgroup>

 <tr>

 <th>Número</th>

 <th>Nombre</th>

 <th>Apellido</th>

 <th>Salario</th>

 </tr>

 <tr>

 <td>0001</td>

 <td>Marta</td>
```

```

<td>Pascual</td>

<td> 1400</td>

</tr>

<tr>

<td>0002</td>

<td>Sandra</td>

<td>Pascual</td>

<td> 1150</td>

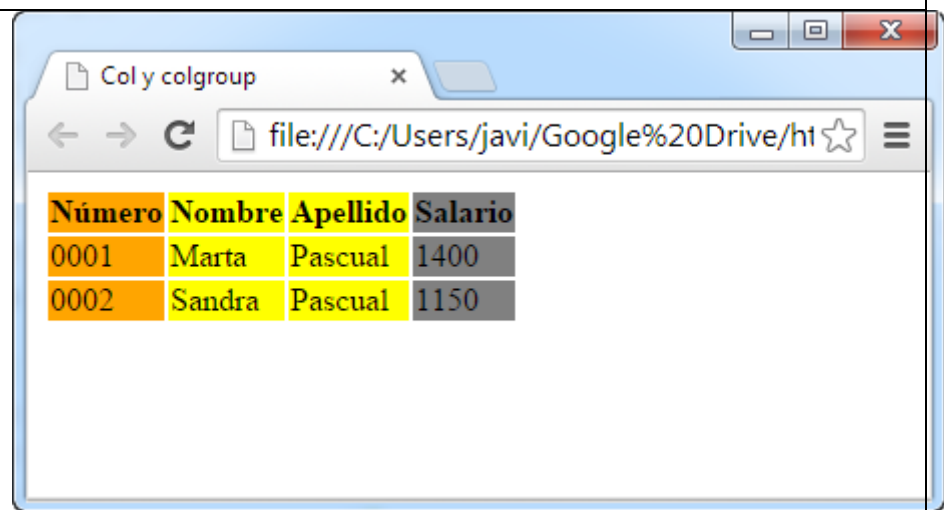
</tr>

</table>

</body>

</html>

```



En este ejemplo se ve que se definen tres grupos de columnas dentro de `<colgroup>`. El primero formado por una columna a la que se aplica color naranja de fondo, el segundo formado por dos columnas (`span=2`) y al que se aplica el color amarillo de fondo y el tercer y último grupo formado por una columna a la que se aplica el color gris de fondo.



*`<colgroup>` y `<col>` se usan principalmente para aplicar estilos a una o varias columnas.*

	<i>Si se aplica un color de fondo o cualquier otro estilo a una celda individual prevalece sobre lo que se defina con &lt;col&gt;</i>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------

Etiquetas	<thead> <tbody> y <tfoot>				
Descripción	<thead> define una o más filas de encabezado en una tabla, <tbody> define una o más filas como sección de una tabla y <tfoot> define una o más filas de pie de la tabla				
Elemento	Bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios					
Diferencia entre html 4.01 y html5	Ninguno de los atributos de HTML4 son soportados en HTML5				

	<i>&lt;thead&gt;, &lt;tbody&gt; y &lt;tfoot&gt; siempre deben ser usados juntos</i>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------

El uso de estas etiquetas permite que si la tabla tiene demasiadas filas que no pueden ser visualizadas a la vez y hay que ir las pasando, entonces la cabecera y el pie se quedan fijos y se ven siempre.

La etiqueta <thead> va detrás de los elementos de tabla como <caption>, <colgroup> o <col> y antes de cualquier <tr>

	<i>En una tabla solo hay una etiqueta &lt;thead&gt; y una &lt;tfoot&gt; aunque puede haber varias &lt;tbody&gt;</i>
-------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------



*<tfoot> se usa antes que <tbody> por lo que el orden que debemos seguir para empelar estas etiquetas sería: <thead>, <tfoot> y <tbody>*

### 2.8.1. Operaciones con tablas

- Unión horizontal de celdas: Con el atributo colspan de la etiqueta <td> se pueden unir celdas contiguas de una misma fila.

Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Fusión columnas</title>

 <meta charset="UTF-8">

 </head>

 <body>

 <table style="border:1px solid #000">

 <tr>

 <td style="border:1px solid #000"> fila 1 celda 1 </td>

 <td style="border:1px solid #000"> fila 1 celda 2 </td>

 <td style="border:1px solid #000"> fila 1 celda 3 </td>

 </tr>

 <tr>

 <td colspan=2 style="border:1px solid #000"> fila 2 celda 1 y 2 </td>

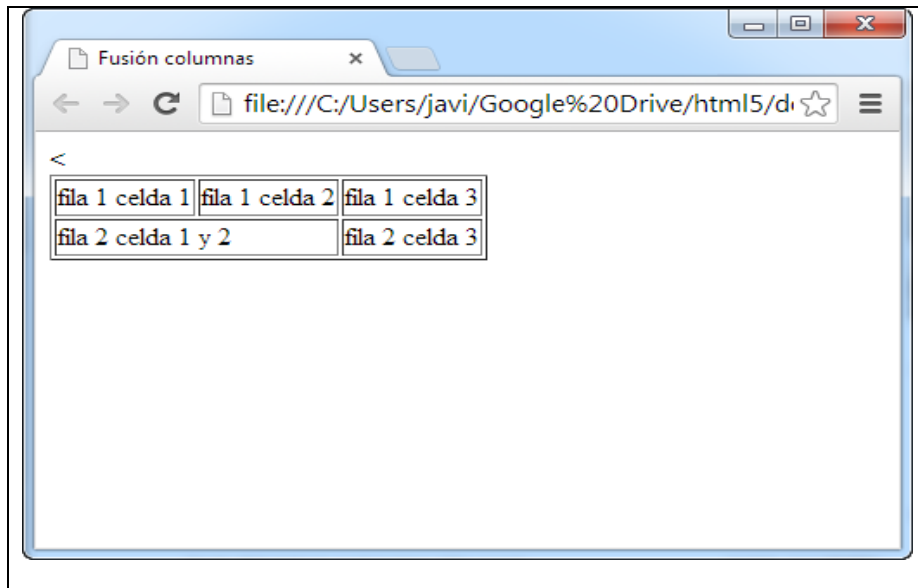
 <td style="border:1px solid #000"> fila 2 celda 3 </td>

 </tr>

 </table>

 </body>

</html>
```



- Unión vertical de celdas: Con el atributo `rowspan` de la etiqueta `<td>` se puede unir celdas contiguas de una misma columna.

Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Fusión filas</title>

 <meta charset="UTF-8">

 </head>

 <body>

 <table style="border:1px solid #000">

 <tr>

 <td rowspan=2 style="border:1px solid #000"> filas 1 y 2 celda
 1 </td>

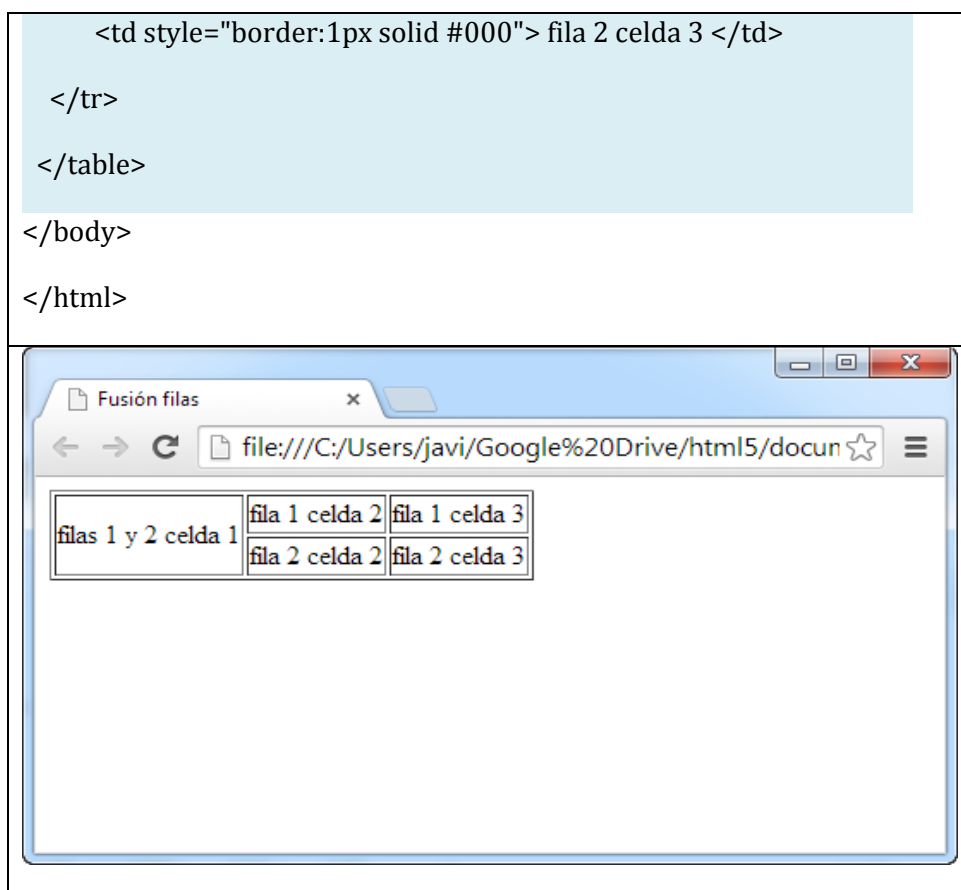
 <td style="border:1px solid #000"> fila 1 celda 2 </td>

 <td style="border:1px solid #000"> fila 1 celda 3 </td>

 </tr>

 <tr>

 <td style="border:1px solid #000"> fila 2 celda 2 </td>
```



*style="border:1px solid #000;"* permite poner un borde a la tabla (elemento `<table>`) o a una celda (elemento `<td>`) de un pixel de grosor y de color negro.

**Ejercicio nº. 10:** Fíjate en la imagen de la tabla inferior y crea una página HTML5 que reproduzca tal cual está dicha tabla. El ancho de la tabla es el ancho de la ventana por lo que usa el siguiente estilo tanto para la tabla como para las celdas:

*style="border:1px solid #000; width:100%;"*

Este estilo dibuja un borde de 1 pixel y que ocupa todo el ancho de la ventana en el caso de la table (`<table>`) y todo el ancho que pueda dependiendo del contenedor en el que está en el caso de la celda (`<td>`). No pienses mucho en cómo funciona el estilo anterior ya que lo veremos más profundamente posteriormente en el curso. Sin embargo, empléalo para que tengas una mejor perspectiva de lo que estás haciendo.

Ten en cuenta también que no se pueden hacer cálculos, es decir, las operaciones son datos que debes poner directamente sin utilizar ninguna fórmula o cálculo. Lo importante de este ejercicio es que identifiques las partes de una tabla y que sepas su utilidad.



Ejemplo de tabla avanzada: x

file:///C:/Users/javi/Google%20Drive/html5/documentacion/ejemplos, ☆ ☰

Gastos mensuales

Producto	Unidades	Precio	Total
Camisetas	10	2	20
Pantalones	3	14	42
Chaquetas	4	20	80
Año 2014		Total	142

Como guía básica fíjate en la siguiente imagen para que veas las partes de la tabla

Ejemplo de tabla avanzada: x

file:///C:/Users/javi/Google%20Drive/html5/documentacion/ejemplos, ☆ ☰

Gastos mensuales

Producto	Unidades	Precio	Total
Camisetas	10	2	20
Pantalones	3	14	42
Chaquetas	4	20	80
Año 2014		Total	142

thead

tbody

tfoot

Caption

El código lo tienes aquí ([enlace 5table.html](#))

**Ejercicio nº. 11:** Realiza la siguiente tabla fusionando filas y columnas. Usaremos también el estilo anterior para que sea más claro el ejercicio.

Fusiones de filas y columnas


Enlace al ejercicio 6table.html

**Ejercicio nº. 12:** Usando todos los conceptos que hemos visto en este tema, crea la tabla de la imagen inferior. Aquí tienes unas pequeñas pautas:

- La tabla ocupa todo el ancho de la página y tiene un borde de color negro de 1 pixel de ancho (style="border:1px solid #000;width:100%;").
- Todas las celdas de la tabla tienen borde de 1 pixel de color negro (style="border:1px solid #000;").
- La cabecera de la tabla, <tbody>, ocupa las dos primeras filas, el pie, <tfoot>, tiene otras dos filas y el cuerpo, <tbody>, el resto.
- Hay una agrupación de columnas (<colgroup>) que permiten dar los colores: orange, olive, aqua y grey a los conjuntos de columnas (<col>).
- Las celdas con contenido en negrita y centrado se definen con la etiqueta <th>.
- Las celdas que no están en negrita se definen con la etiqueta <td>.
- Para centrar un texto en una celda <td> se usa el estilo *text-align:center*.

Horario laboral semanal

Días laborables de la semana					
	Lunes	Martes	Miércoles	Jueves	Viernes
Mes de octubre	8,30	9,00	8,45	10,00	9,45
	9,00		9,00	10,15	10,15
	9,30		9,15	10,30	
	10,00	9,30			
	Lunes	Martes	Miércoles	Jueves	Viernes
Días laborables de la semana					

## 2.9. LISTAS

Una lista es un párrafo estructurado que contiene una serie de elementos HTML que tienen más significado de forma conjunta. Los tipos de listas son:

- **Listas no ordenadas:** Este tipo de lista se usa para enumerar elementos que no tengan un orden definido. Su etiqueta de definición es `<ul>` y todo lo que se incluya dentro a partir de esa etiqueta y la de cierre `</ul>`, será el contenido de la lista con los componentes de la misma dentro de etiquetas `<li>`. También puede contener un título de lista definido por la etiqueta `<lh>`

Etiqueta						
Descripción	Define una lista no ordenada					
Elemento	Bloque					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos					
Atributos propios						
Diferencia entre html 4.01 y html5	Los atributos compact y type están suprimidos en HTML5					

Etiqueta						
Descripción	Define un elemento de una lista ordenada, no ordenada o menú					
Elemento	Bloque					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos					
Atributos propios	value					
Diferencia	El atributo "type" está suprimido en HTML5 y el atributo "value" que está					

entre html 4.01 y html5	obsoleto en HTML4.01 pero es soportado en HTML5
-------------------------	-------------------------------------------------

En la siguiente tabla están los atributos propios de <li>

Atributos propios	Valor	Descripción
value 	Número	Especifica el valor de la lista en ese elemento. Solo para listas ordenadas (  )

	<i>&lt;li&gt; define elementos de listas ordenadas, &lt;ol&gt;, no ordenadas, &lt;ul&gt; y menú, &lt;menu&gt;</i>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------

	<i>Usa estilos CSS para definir el tipo de lista</i>
-----------------------------------------------------------------------------------	------------------------------------------------------

Ejemplo:

```

<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Col y colgroup</title>

 <meta charset="UTF-8">

</head>

<body>

 <lh> Título de la lista </lh>

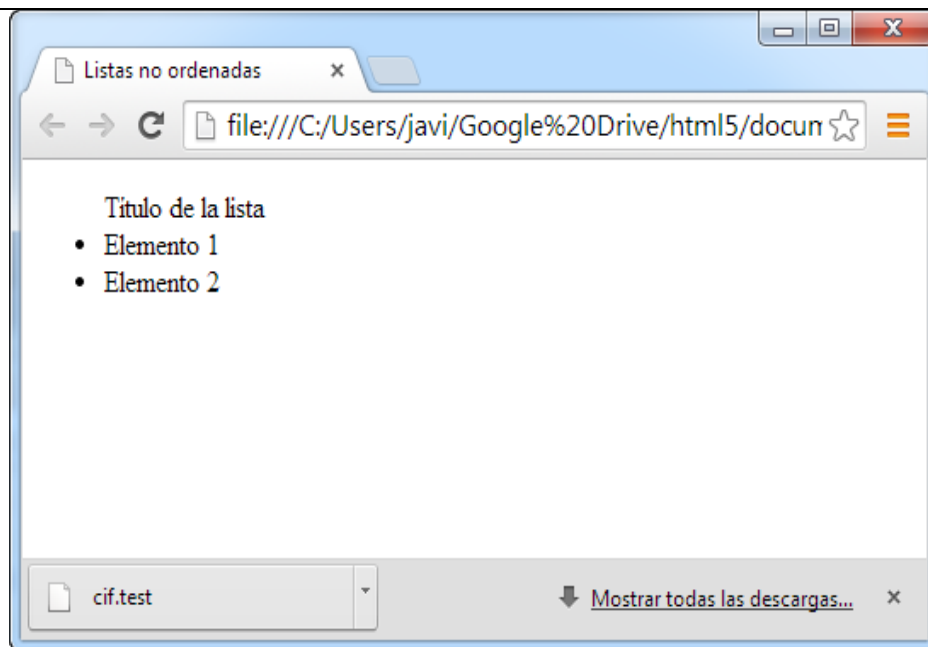
 Elemento 1

 Elemento 2


```

```
</body>
```

```
</html>
```



- **Listas ordenadas:** En este tipo de lista se usa un número o letra para indicar el orden de cada elemento de la lista. Su etiqueta de definición es `<ol>` y su uso es idéntico a `<ul>`



*La listas ordenadas no se ordenan bajo ningún criterio, es decir, solamente siguen el orden de la lista independientemente de su contenido*

Etiqueta						
Descripción	Lista ordenada					
Elemento	Bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					
Atributos	Reversed, start, type					

propios	
Diferencia entre html 4.01 y html5	Los atributos “start” y “type” de HTML 4.01 están obsoletos, pero los soporta HTML5.   El atributo “reversed” es nuevo en HTML5   El atributo “compact” está suprimido en HTML5

Atributos propios	Valor	Descripción
reversed     	reversed	Especifica que el orden de la lista debería ser descendente. No soportado por Safari 5 y versiones anteriores
start    	número	Especifica el primer valor de la lista
type    	1 A a I i	Especifica el tipo de marcador que se usa en la lista.

 → Nuevo en HTML5

Ejemplo:

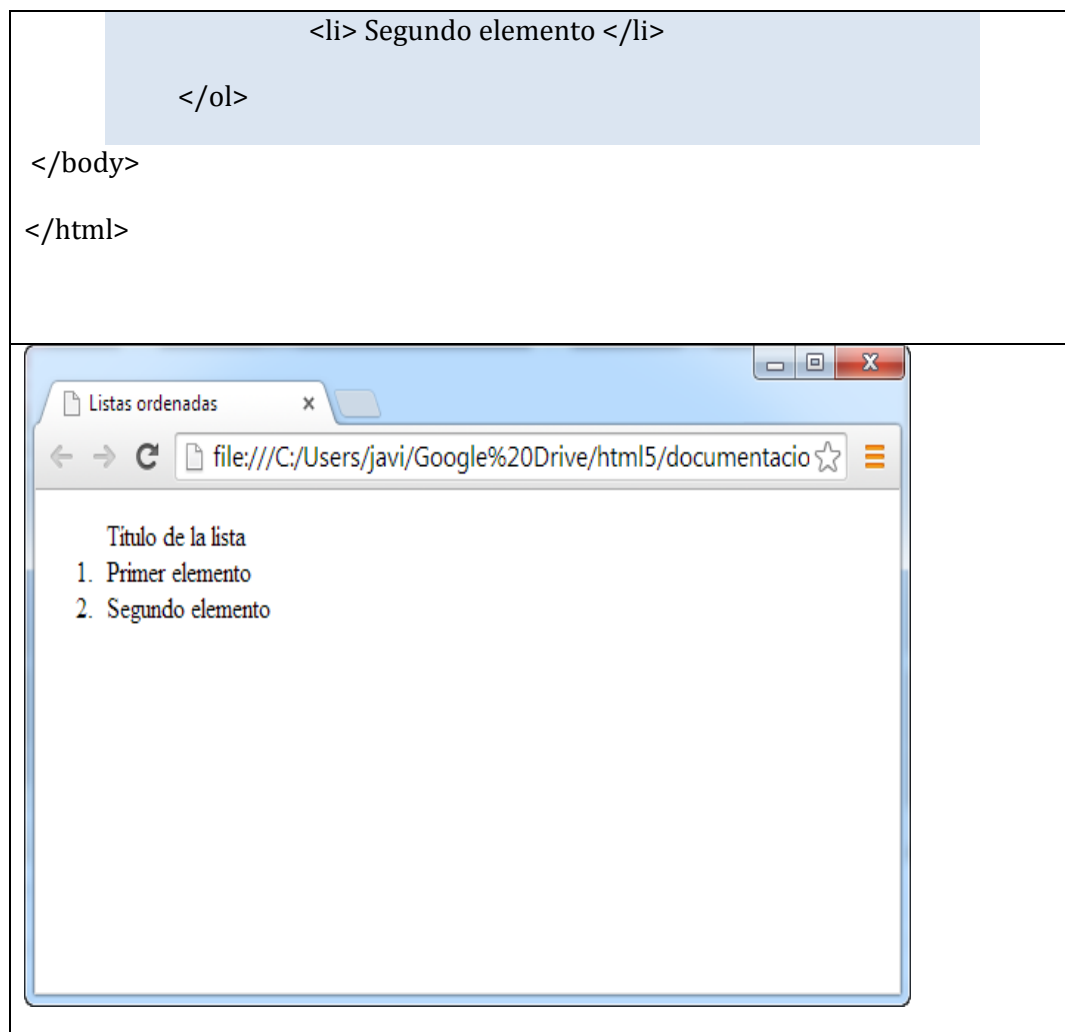
```

<!DOCTYPE html>
<html lang="es-es"
 <head>
 <title> Listas Ordenadas</title>
 <meta charset="UTF-8">
 </head>
 <body>

 <lh> Título de la lista </lh>
 Primer elemento

 </body>
</html>

```



- **Listas de descripción:** En este tipo de lista se indican uno o varios términos y se dan su definiciones. Su etiqueta principal es `<dl>` y entre ella y la de cierre `</dl>` se indica el término con la etiqueta `<dt>` y se define con una o varias etiquetas `<dd>`

Etiqueta	<dl>				
Descripción	Define una lista de descripción				
Elemento	Bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios					

Diferencia entre html 4.01 y html5	En HTML 4.01 define una lista de definición y en HTML5 define una lista de descripción
------------------------------------	----------------------------------------------------------------------------------------

Etiqueta	<dt>				
Descripción	Se emplea para indicar los términos de una lista de descripción				
Elemento	Bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios					
Diferencia entre html 4.01 y html5	En HTML 4.01 define los términos de una lista de definición y en HTML5 define los términos de una lista de descripción				

Etiqueta	<dd>				
Descripción	Define los términos de una lista de descripción				
Elemento	Bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios					
Diferencia entre html 4.01 y html5	En HTML 4.01 define los términos de una lista de definición y en HTML5 define los términos de una lista de descripción				



Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es"

 <head>

 <title> Listas Ordenadas</title>

 <meta charset="UTF-8">

 </head>

 <body>

 <dl>

 <dt> Gato

 <dd> Mámifero que caza ratones

 <dd> Herramienta mecánica

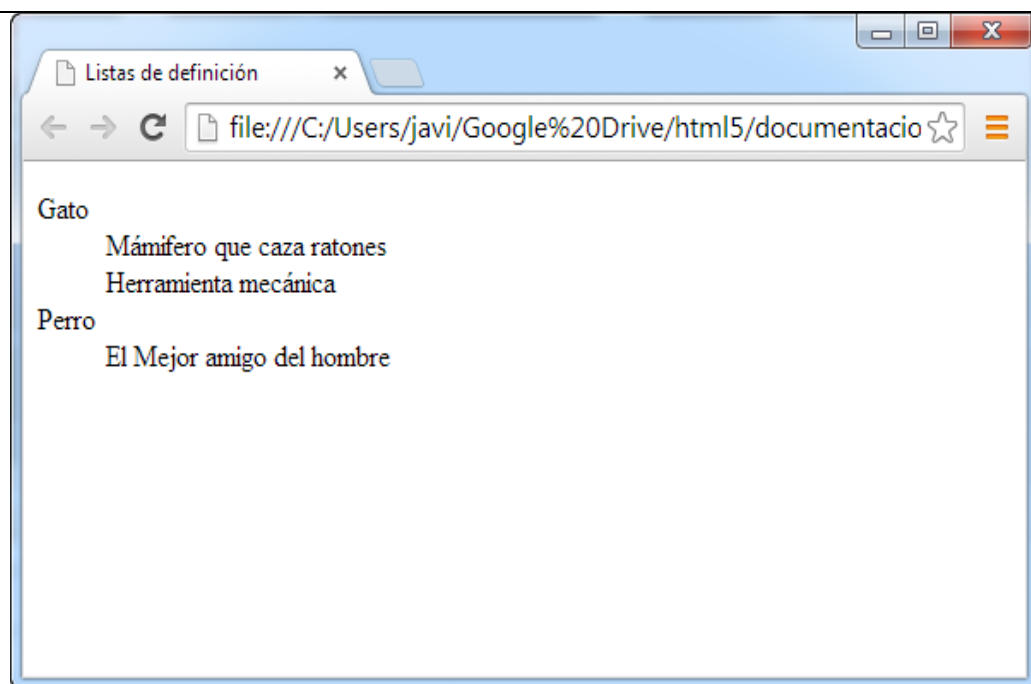
 <dt> Perro

 <dd> El Mejor amigo del hombre

 </dl>

 </body>

</html>
```



### 2.9.1. Anidamiento de listas

Una lista puede tener definidas otras listas de cualquier tipo dentro de alguno de sus elementos entonces hablamos de listas anidadas y de niveles, donde nivel es la profundidad de una lista o el número de listas que tiene por encima alguna de ellas.

Ejemplo de lista de nivel 3

```
<!DOCTYPE html>

<html lang="es-es"

<head>

 <title> Listas Ordenadas</title>

 <meta charset="UTF-8">

</head>

<body>

 <lh> Provincias, comarcas y pueblos </lh>

 Soria

 Pinares

 Covalada

 Vinuesa

 El Valle

 El Royo

 Molinos de Razón


```

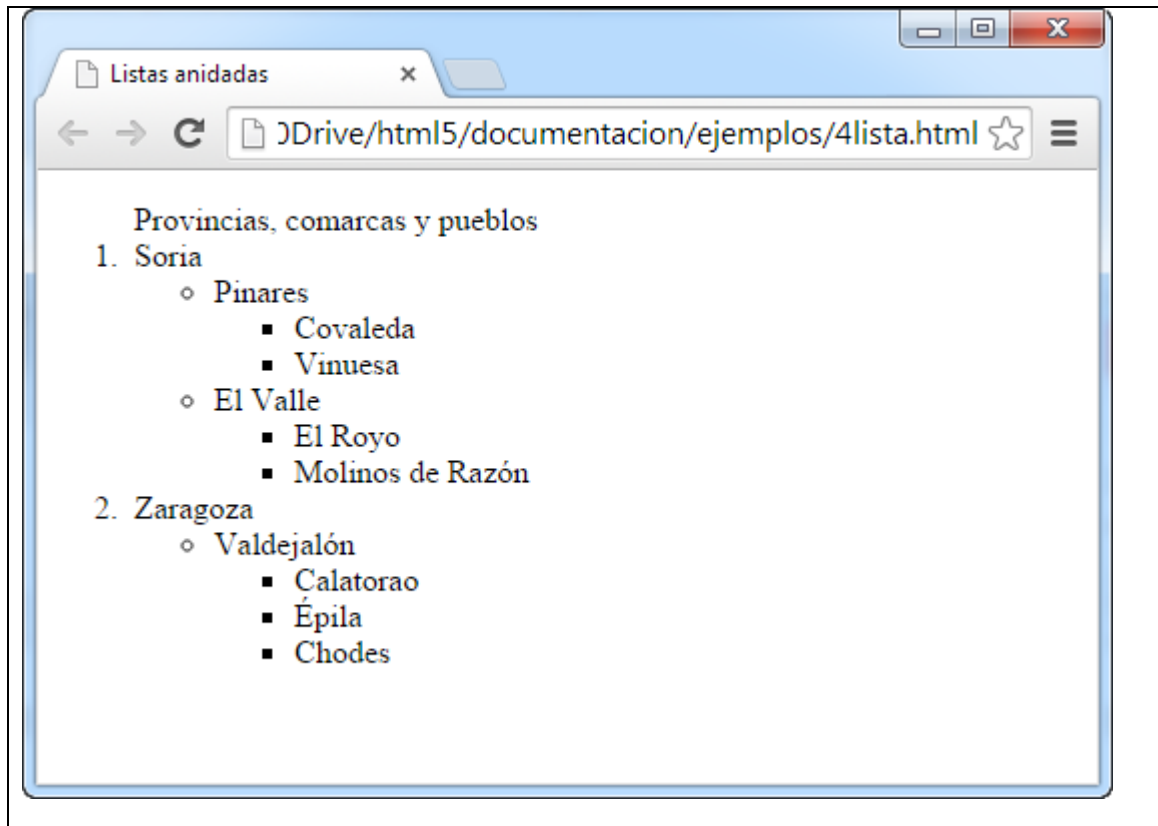
```


 Zaragoza

 Valdejalón

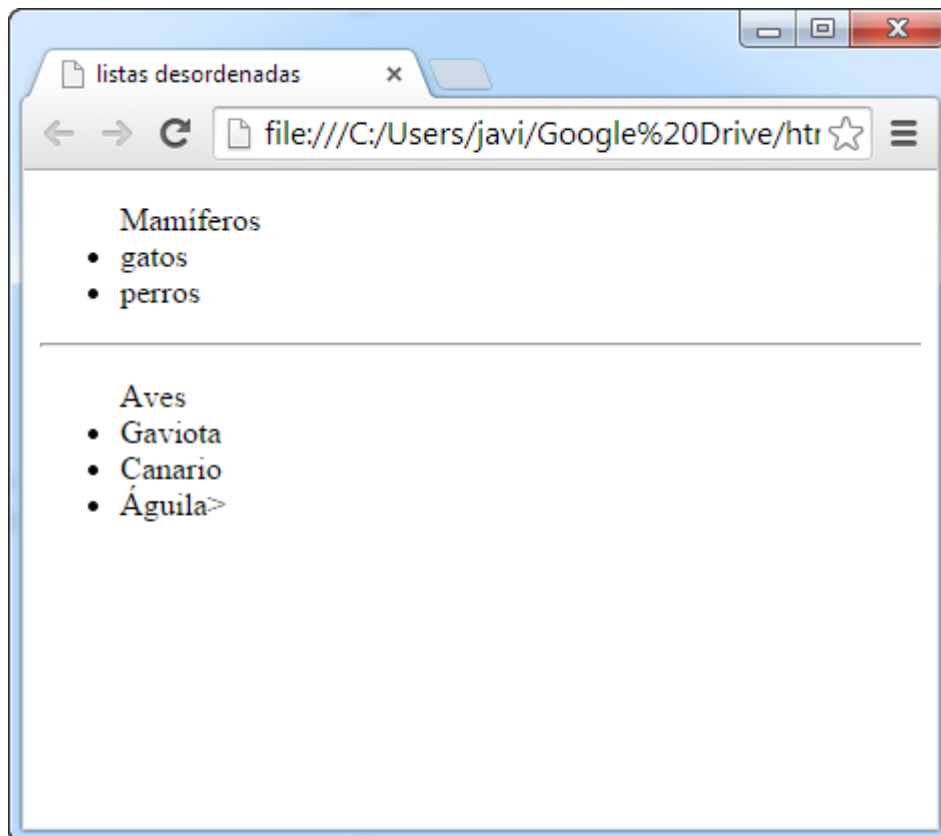
 Calatorao
 Épila
 Chodes

</body>
</html>
```



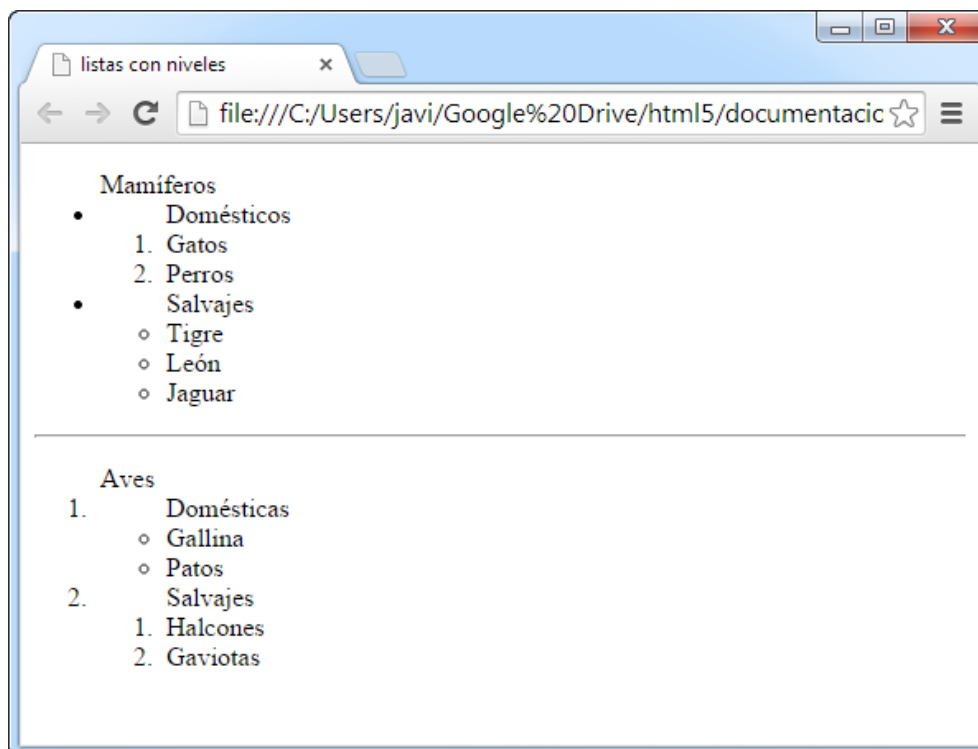
En este ejemplo Covalada, Vinuesa, Calatorao, etc. son listas de nivel 3 ya que tienen dos niveles por encima de ellas, Pinares, El Valle y Valdejalón son listas de nivel 2 y Soria y Zaragoza son de nivel 1

**Ejercicio 13:** Realiza el siguiente ejercicio con dos listas desordenadas



Para separar ambas listas usa la etiqueta `<hr>` que dibuja una línea horizontal

**Ejercicio 14:** Fíjate en la imagen e intenta hacerla tal cual identificando bien los tipos de listas, sus títulos y sus niveles.

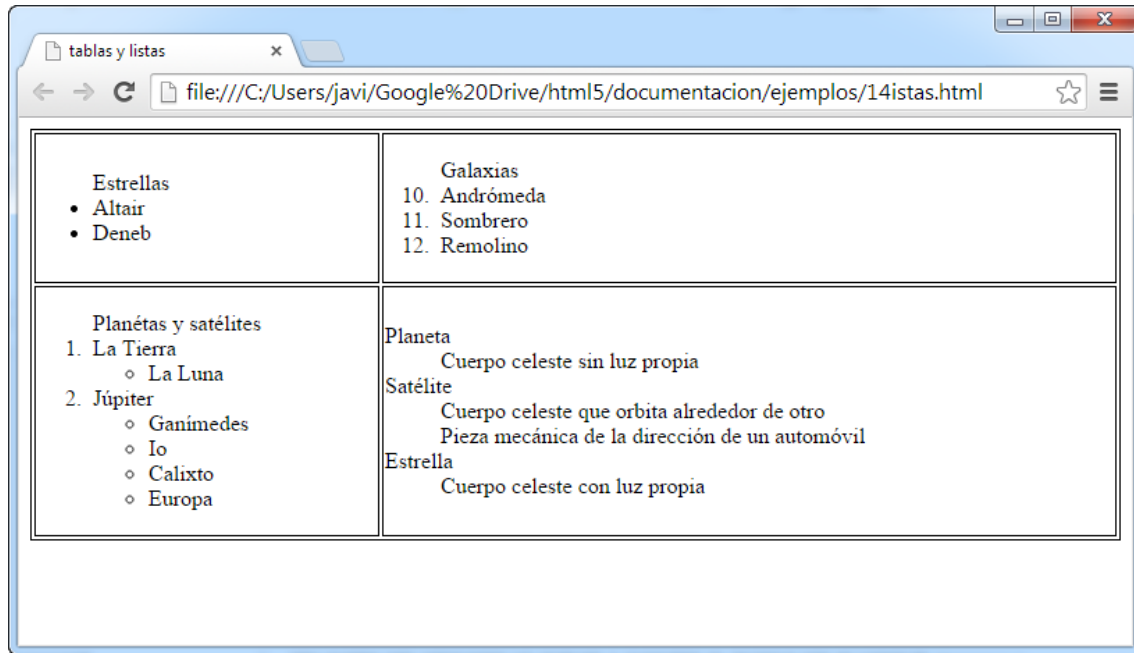


**Ejercicio 15:** Utilizando una tabla de dos filas y dos columnas realiza el siguiente ejercicio empleando el siguiente estilo para la tabla

`style="width:100%; border:1px solid #000;"`

y este para cada celda

`style="border:1px solid #000;"`



Estrellas       • Altair     • Deneb	Galaxias       10. Andrómeda     11. Sombrero     12. Remolino
Planetas y satélites       1. La Tierra      ◦ La Luna           2. Júpiter      ◦ Ganímedes     ◦ Io     ◦ Calixto     ◦ Europa	Planeta   Cuerpo celeste sin luz propia   Satélite   Cuerpo celeste que orbita alrededor de otro   Pieza mecánica de la dirección de un automóvil   Estrella   Cuerpo celeste con luz propia

## 2.10. ENLACES

Los enlaces, vínculos o hipervínculos se utilizan para establecer relaciones entre dos objetos que pueden ser páginas web, imágenes, documentos, etc., es decir, que pueden ser archivos.

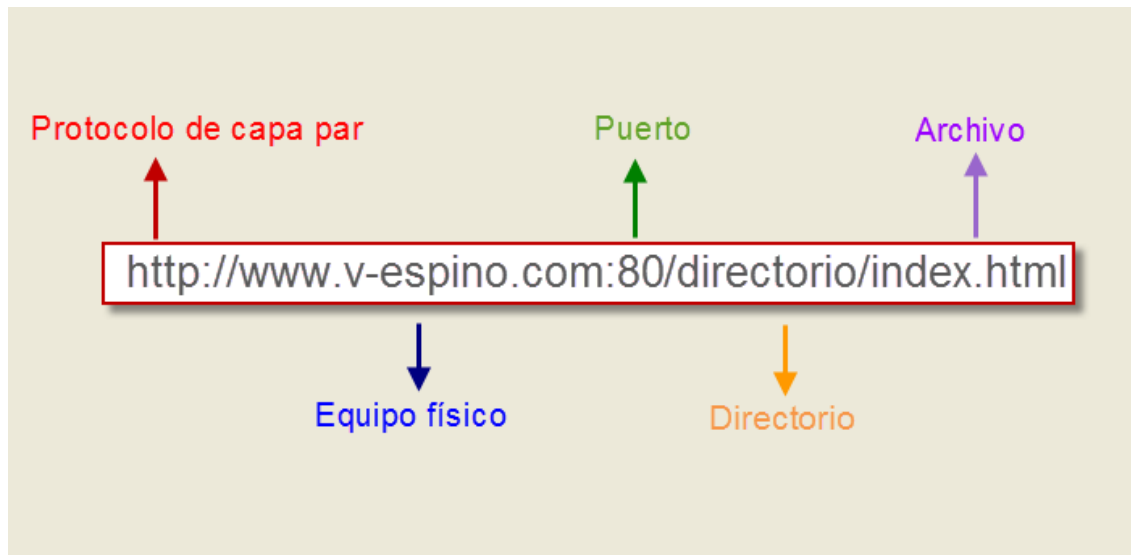
Una característica importante de los enlaces es que están formados por un origen, un destino y una dirección. En otras palabras, el enlace comienza en un objeto origen y apunta hacia otro, que es el destino, iniciando el acceso a este último a través de un clic de ratón sobre el enlace.

### 2.10.1. URL (localizador uniforme de recursos).

URL es un texto que cumple unas normas muy concretas y que sirve para referenciar o identificar un recurso así como localizar el equipo de una red de comunicaciones como, por ejemplo, Internet donde se aloja dicho recurso.

La URL de un objeto tiene dos fines importantes:

- Identificarlo y diferenciarlo claramente de los demás objetos.
- Localizarlo de la forma más rápida a través de la información que suministra la propia URL.



En la imagen anterior vemos un ejemplo de URL que cumple la normativa TCP/IP que es la usada en Internet. Las URL están formadas por las siguientes partes:

- Protocolo de capa par: Normas que debe cumplir el navegador para visualizar sin errores la información del archivo. Otros protocolos de capa par son ftp, DNS, https, pop3, etc.

- Equipo físico: Equipo en el que se encuentra alojado el archivo al que se quiere acceder. Está representado por un nombre DNS o por una dirección ip.
- Puerto: Identificador dentro del sistema de la aplicación servidora que ofrece el archivo.
- Directorio: Directorio donde se encuentra el archivo y que forma parte de la estructura que controla la aplicación servidora que la ofrece.
- Archivo: Fichero al que referencia la URL.

#### - URL absolutas.

Son las que definen todas las partes de la URL por lo que se tiene todo lo necesario para acceder al objeto destino.

Ejemplo:

<http://v-espino.com/index.html>



*Las URL absolutas tienen el inconveniente de que si el recurso cambia de ubicación todas las URL hay que cambiarlas también*

#### - URL relativas.

Son las que no definen todas las partes de la URL con el fin de hacerlas más simples. En realidad son URL incompletas que se crean a partir de URL absolutas y que obligan a conocer la URL origen del enlace para que funcionen sin problemas.

Ejemplo:

/index.html

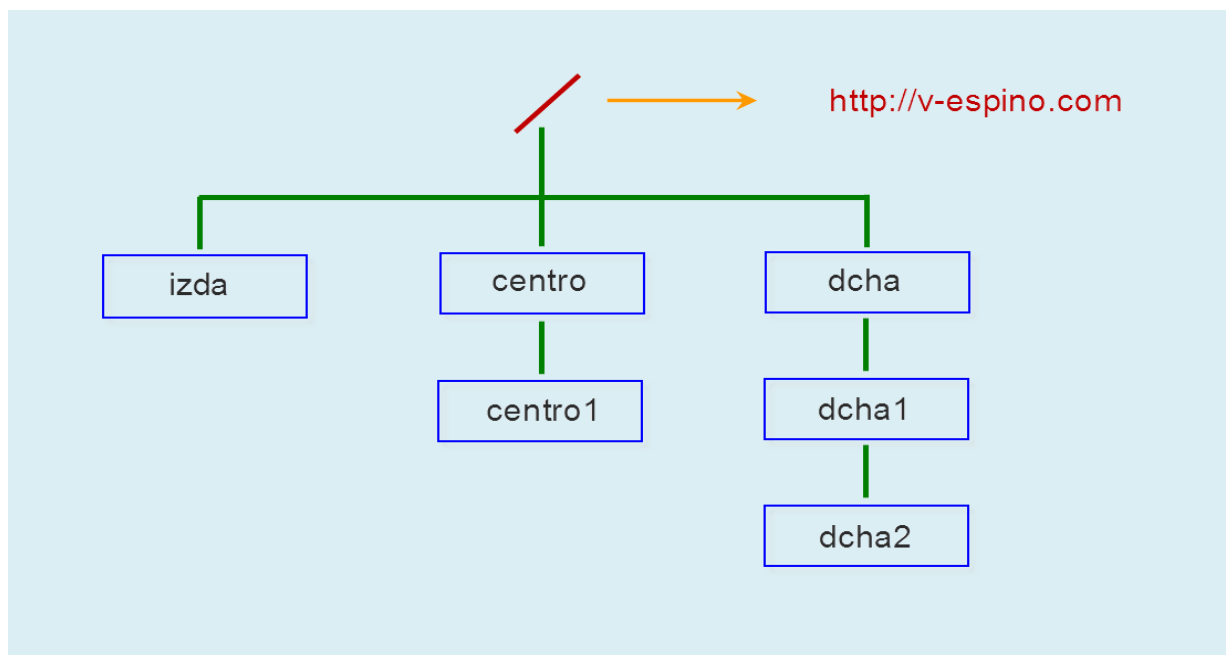
Como vemos esta URL no es muy significativa ya que no sabemos en qué ubicación está el archivo. Por eso es importante conocer que la URL absoluta del ejemplo es <http://v-espino.com/index.html> lo que indica que index.html está en la raíz del servidor v-espino.com.



*Las URL relativas sirven para referirse a recursos dentro del mismo ordenador que pertenecen a la estructura normalmente de una aplicación servidora*

Fíjate en la imagen siguiente y supón que tenemos dos archivos uno llamado origen.html y otro llamado destino.html y que origen.html tiene un enlace que va a destino.html. Vamos a ponerlos en diferentes partes de la estructura y vamos a ver como quedarían las URL relativas y absolutas cuando se usan en el archivo origen.html para referirse a destino.html





Carpeta de origen.html	Carpeta de destino.html	URL ABSOLUTA de destino.html con respecto a la carpeta de origen.html
/	/	http://v-espino.com/destino.html
/izda	/izda	http://v-espino.com/izda/destino.html
/izda	/centro	http://v-espino.com/centro/destino.html
/centro/centro1	/dcha	http://v-espino.com/dcha/destino.html
/dcha/dcha1	/centro/centro1	http://v-espino.com/centro/centro1/destino.html
/	/dcha/dcha1/dcha2	http://v-espino.com/dcha/dcha1/dcha2/destino.html

Como se ve en la tabla anterior la URL absoluta de destino.html solo depende de su ubicación independientemente de donde esté origen.html.

Si llevamos las páginas a otro servidor que no sea v-espino.com entonces no funcionará ninguna de la URL anteriores.

Antes de pasar a las URL relativas hay que saber que “..” representa la carpeta inmediatamente superior y no hace falta usar el nombre de dicha carpeta ya que una carpeta solo puede tener otra por encima a excepción de la carpeta / o raíz que, evidentemente, no tiene ninguna.

Carpeta de origen.html	Carpeta de destino.html	URL RELATIVA de destino.html con respecto a la carpeta de origen.html
------------------------	-------------------------	-----------------------------------------------------------------------

/	/	destino.html
/izda	/izda	destino.html
/dcha/dcha1/dcha2	/dcha/dcha1/dcha2	destino.html
/izda	/	../destino.html
/izda	/centro	../centro/destino.html
/izda	/centro/centro1	../centro/centro1/destino.html
/izda	/dcha	../dcha/destino.html
/izda	/dcha/dcha1/dcha2	../dcha/dcha1/dcha2/destino.html
/centro/centro1	/	../../destino.html
/centro/centro1	/izda	../../izda/destino.html
/centro/centro1	/dcha/dcha1	../../dcha/dcha1/destino.html
/centro/centro1	/centro	../destino.html
/dcha/dcha1/dcha2	/	../,../destino.html
/dcha/dcha1/dcha2	/izda	../,../izda/destino.html
/dcha/dcha1/dcha2	/centro/centro1	../,../centro/centro1/destino.html
/dcha/dcha1/dcha2	/dcha	../,../destino.html
/dcha/dcha1/dcha2	/dcha/dcha1	../destino.html
/	/dcha/dcha1/dcha2	dcha/dcha1/dcha2/destino.html
/dcha	/dcha/dcha1/dcha2	dcha1/dcha2/destino.html
/dcha/dcha1	/dcha/dcha1/dcha2	cha2/destino.html

Las conclusiones que se sacan de la tabla anterior son:

- Si ambas páginas están en la misma carpeta con solo poner el nombre de la página de destino vale.
- Si la página destino está justo en la carpeta superior de la del origen con poner ../nombre de la página de destino vale.
- Si la página destino está dos carpetas por encima de la de origen con poner ../,../nombre de la página de destino vale.
- Si la página destino está una carpeta por debajo de la de origen con poner el nombre de la carpeta seguido del nombre de la página destino vale.
- Si la página destino está varias carpetas por debajo de la de origen con poner las carpetas sucesivamente y luego el nombre de la página destino vale.

- Si la página destino se encuentra en una carpeta que no está ni encima ni debajo de la de origen se ponen tantos “..” como carpetas hay hasta la raíz y luego se van poniendo los nombres de las carpetas intermedias a medida que se va descendiendo hasta la carpeta de la página de destino seguido del nombre de la página destino.
- Si llevamos las páginas a otro servidor que no sea v-espino.com las URL serían perfectamente válidas siempre que dicho servidor tenga una estructura igual de carpetas.

### 2.10.2. Etiqueta <a>

Para crear un enlace de cualquier tipo se emplea la etiqueta <a> cuya principal función es especificar la dirección del destino ya sea en la propia página (enlace interno) o fuera de la misma (enlace externo).

Etiqueta	<a>				
Descripción	Crea un enlace a un objeto				
Elemento	En línea				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios	href, hreflang, rel, target, download, type				
Diferencia entre html 4.01 y html5	Los atributos compact, name y de aspecto están suprimidos en HTML5 y en los enlaces internos se usa el atributo global <i>id</i> (HTML5) en vez de <i>name</i> (HTML4)				

Atributos propios	Valor	Descripción
download      	Nombre de fichero	Especifica que el fichero indicando será descargado cuando se pulse sobre el enlace
href     	URL	Especifica la URL a la que apunta el enlace
hreflang     	Idioma	Especifica el lenguaje del documento destino

<b>media</b> 	<b>Dispositivo</b>	Especifica cual es el dispositivo óptimo para usar con el documento enlazado
<b>rel</b> 	alternate author bookmark help license next nofollow norereferrer prefetch prev search tag	Especifica la relación entre el documento origen y el documento enlazado
<b>target</b> 	_blank _parent _self _top Nombre marco	Especifica dónde se abrirá el documento enlazado
<b>type</b> 	<b>Tipo MIME</b>	Especifica el tipo MIME del documento enlazado

 → Nuevo en HTML5

	<i>El atributo más importante de &lt;a&gt; es <b>href</b> ya que indica la URL del fichero destino en los enlaces externos o la URL más el ancla o marcador en los enlaces internos.</i>
-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

En general los enlaces tienen la siguiente estructura

**<a href="URL">Texto del enlace </a>**

Donde URL es la dirección del enlace y "Texto del enlace" es lo que se visualizará en el navegador subrayado normalmente en azul y que se pulsa para ir al destino.

	<i>Se puede emplear una imagen o cualquier elemento HTML5 en vez de texto para hacer clic sobre él en un enlace.</i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------

**Ejemplo:**

```
!DOCTYPE html>

<html>

<head>

 <title>Enlace básico</title>

 <meta charset="UTF-8">

</head>

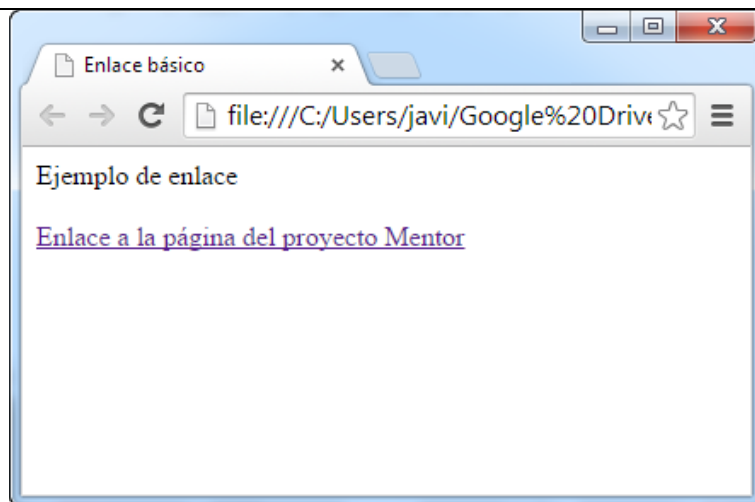
<body>

 Ejemplo de enlace

 Enlace a la página del
 proyecto Mentor

</body>

</html>
```



En este ejemplo si pulsamos sobre “Enlace a la página del proyecto Mentor” abrirá la página correspondiente a la dirección <http://www.mentor.mec.es>.

Ejemplo:

```
!DOCTYPE html>

<html>

<head>

 <title>Enlace básico con imagen</title>
```

```
<meta charset="UTF-8">

</head>

<body>

 Ejemplo de enlace

</body>

</html>
```



En este ejemplo se hace clic sobre la imagen `mentorhtml5.png` que se encuentra en la misma carpeta que el documento que tiene el enlace y nos abre la página del Proyecto Mentor (<http://www.mentor.mec.es>).

Por defecto, los enlaces pueden aparecer en diferentes colores según hayan sido utilizados o vayan a serlo.

- Un enlace que no se haya pulsado, o sea, que no haya sido visitado, estará subrayado y de color azul.
- Un enlace pulsado, o sea, visitado, estará subrayado y de color púrpura.
- Un enlace activo o que acaba de ser pulsado estará subrayado y de color rojo.



*Se pueden cambiar los anteriores colores a través de estilos CSS*

El destino del enlace determina en qué ventana va a ser abierta la página enlazada y se especifica a través del parámetro *target*. Sus valores son

Valor de target	Descripción
_blank	Abre la página enlazada en una nueva ventana o pestaña del navegador
_self	Abre la página enlazada en la misma ventana que la página que contiene el enlace (es la opción por defecto)
_parent	Abre la página enlazada en el marco padre de la página que contiene el enlace
_top	Abre la página enlazada en la ventana completa del navegador
Nombre de marco	Abre la página enlazada en un marco concreto

Prueba en alguno de tus navegadores el siguiente ejemplo para que veas cómo funciona la visualización de enlaces. Los valores de *target* referidos a marcos no los vamos a ver porque los marcos están en desuso.

```
!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
 <title>Enlace básico</title>
```

```
 <meta charset="UTF-8">
```

```
</head>
```

```
<body>
```

```
 Ejemplo de enlace


```

```
 Enlace a la página
 del proyecto Mentor

```

```
 Enlace a la página del
 IES Virgen del Espino

```

```
Enlace a la página de la
Del consorcio w3c
```

```
</body>
```

```
</html>
```



*Una URL siempre apunta a un archivo en un servidor. Si se omite el nombre del fichero normalmente el archivo al que se refiere se llama index.html, index.htm o index.php entre otros (depende de la configuración del servidor donde esté alojado dicho fichero)*

#### - Enlace externo.

Es el enlace que va a una página u objeto que se encuentra fuera del propio servidor o equipo donde está la página que lo referencia.

#### Ejemplo:

```
 Enlace a la página del proyecto
mentor
```

Este tipo de enlaces no tienen que ser siempre a páginas web sino que pueden servir para abrir clientes como los de correo (en este caso servirían para enviar un e-mail) o referencias a archivos para poderlos descargar.

#### Ejemplos:

```
Manda un e-mail a esa dirección
de correo
```

En este caso se abrirían el cliente de correo que tenemos configurado por defecto (Outlook express, kmail, etc.) con la opción de mensaje nuevo y en el destinatario la dirección que tenemos en el enlace.

```
Descargar este archivo
```

En este ejemplo al pulsar "Descargar este archivo" se inicia el proceso de descarga del archivo.zip desde el servidor a nuestro equipo

```

```



```
Descarga archivo por ftp
```

En este ejemplo al pulsar “Descarga archivo por ftp” se inicia el proceso de descarga anónima de archivo.zip desde el servidor ftp que está corriendo en el equipo ftp.v-espino.com

- **Enlace interno.**

Estos enlaces nos llevan a una parte determinada de un documento donde hay definida un ancla o marcador por lo que la URL se sustituye por el nombre del ancla o marcador precedida por el símbolo “#”.

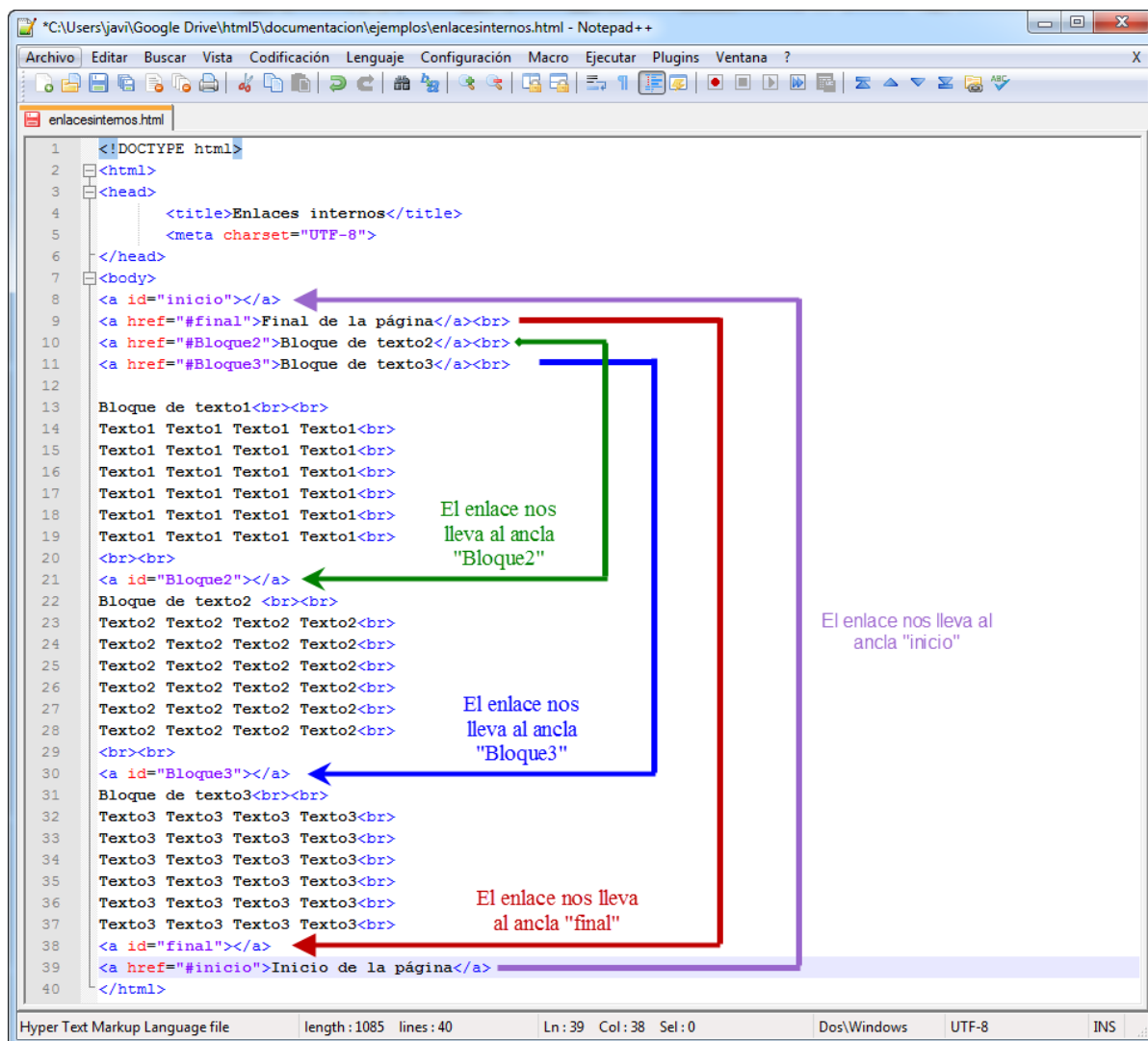
Se utilizan cuando un documento html es tan grande que no se puede visualizar toda la información por lo que podemos marcar una posición dentro de dicho documento y acceder a ella a través de un enlace.

**Ejemplo:**

```
Ir al marcador
```

El ancla se coloca en el lugar donde queremos que vaya el enlace con la etiqueta <a id="nombre del ancla"></a>

**Ejemplo completo:**



También se pueden crear enlaces internos que apunten a partes de otros documentos por lo que un enlace de este tipo abriría el archivo del enlace y nos situaría en el lugar donde esté definida el ancla.

### Ejemplo:

En este ejemplo suponemos que esta línea está dentro de un documento html que no es <http://www.mentor.mec.es/index.html>, por lo que, cuando se pulsa el enlace, se abre la página anterior y se posiciona en el ancla “ejercicios” que hay en ese archivo

```
 Enlace a la parte ejercicios de la página index.html
```

## 2.11. Imágenes

Las imágenes son uno de los elementos que más se emplean en las páginas web debido a que les da un mejor aspecto que si fueran solo texto. Existen dos tipos de imágenes:

- De contenido: Proporcionan información y complementan el texto de una página. Se incluyen directamente en la página con la etiqueta <img>
- Estéticas: Se emplean para mejorar el aspecto estético de la página y suelen insertarse a través de atributos de etiquetas u hojas de estilos CSS. Un ejemplo sería el fondo de un documento html.

	<i>Las imágenes mejoran la calidad de una página web pero hay que tener cuidado con los aspectos estéticos a la hora de colocarlas en nuestros documentos html.</i>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Etiqueta	<img>				
Descripción	Inserta una imagen				
Elemento	In line				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios	Alt, Height, crossorigin, src, ismap, usermap, width				
Diferencia entre html 4.01 y html5	Los atributos align, border, hspace, longdesc and vspace no están soportados en HTML5 pero es de los pocos elementos en el que se pueden usar los atributos height y width sin tener que utilizar estilos CSS				

Atributos propios	Valor	Descripción
alt 	texto	Especifica el texto alternativo a la imagen si no se visualiza correctamente
crossorigin 	anonymous use-credentials	Permite el uso de imágenes de sitios de terceros en <canvas>
height 	pixels	Especifica la altura de la imagen
ismap	ismap	Especifica que es un mapa de imagen

		gestionada por el servidor e implica que <img> debe estar dentro de un enlace
src 	URL	Especifica la URL de la imagen
usemap 	#nombremapa	Especifica que la imagen es un mapa del lado cliente
width 	pixels	Especifica el ancho de la imagen

 → Nuevo en HTML5

Los únicos atributos obligatorios son src y alt aunque es muy recomendable especificar el tamaño de la imagen a visualizar bien por estilos o bien por los atributos width y height para poder organizar mejor los contenidos de la página web.

	<img> es de los pocos elementos HTML5 en la que existe la posibilidad de manipular el ancho y la altura de la imagen con los atributos width y height ya que, en las demás, esto solo se hace con estilos CSS.
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<img> es una etiqueta vacía, es decir, no tiene etiqueta de cierre
-------------------------------------------------------------------------------------	--------------------------------------------------------------------

El formato general y más básico que tiene un elemento <img> es el siguiente:



Donde URL cumple todo lo visto en el tema de enlaces e identifica y localiza un archivo imagen.

HTML no impone condiciones sobre el formato gráfico que se puede emplear en las imágenes, por lo que <img> puede incluir cualquier tipo de imagen. No obstante, hay que tener en cuenta que depende de los navegadores el que se pueda mostrar o no una imagen por lo que se recomienda usar los formatos gráficos GIF, JPEG y PNG para que no haya problemas con su visualización.

	Hay que tener cuidado con el tamaño real de la imagen que se va a insertar en un documento HTML ya que si dicha imagen es muy grande ralentizará la carga de la página en el navegador
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



La unidad de tamaño de una imagen es el pixel o punto de color del monitor.

### Ejemplo:

```
<!DOCTYPE html>
```

```
<html lang="es-es">
```

```
<head>
```

```
<title>Imagen</title>
```

```
<meta charset="UTF-8">
```

```
</head>
```

```
<body>
```

```

```

```
</body>
```

```
</html>
```

```
<!DOCTYPE html>
```

```
<html lang="es-es">
```

```
<head>
```

```
<title>Imagen</title>
```

```
<meta charset="UTF-8">
```

```
</head>
```

```
<body>
```

```

```

```
</body>
```

```
</html>
```



En este ejemplo, usamos la versión con estilos o la que no los tiene el resultado es el mismo, es decir, acorta la visualización de la imagen a un ancho de 80 pixels y a una altura de 50 pixels. Esta forma de visualizar una imagen nunca modifica el tamaño real de la imagen que ocupará lo mismo independientemente de los valores de ancho y largo.

Permite visualizar un documento HTML dentro de un área de otro documento HTML.

Etiqueta	<iframe>				
Descripción	Visualiza un documento HTML dentro de un área de otro documento HTML				
Elemento	En línea				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios	name, sandbox, seamless, src, srcdoc, width y height				
Diferencia	Los atributos align, frameborder, longdesc, marginheight, marginwidth y				

entre html 4.01 y html5	scrolling no están soportados en HTML5 pero es de los pocos elementos en el que se pueden usar los atributos height y width sin tener que utilizar estilos CSS
-------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------

Atributos propios	Valor	Descripción
height 	pixels	Especifica la altura del elemento
name 	texto	Especifica el nombre de un marco
sandbox  	"" Allow-forms Allow-same-origin Allow-scripts Allow-top-navigation	Indica un conjunto de restricciones extras para el contenido del marco
seamless  	seamless	Especifica si el contenido del marco debe parecer como una parte de la página contenedora sin bordes ni barras de scroll
src 	URL	Especifica la URL de la página a visualizar en el marco
srcdoc  	Código html	Especifica código html para ser visualizado dentro del marco
width 	pixels	Especifica el ancho del marco

 → Nuevo en HTML5

### Ejemplo básico:

```
<!DOCTYPE html>

<html>

 <head>
```

```
<title>Iframes</title>

</head>

<body>

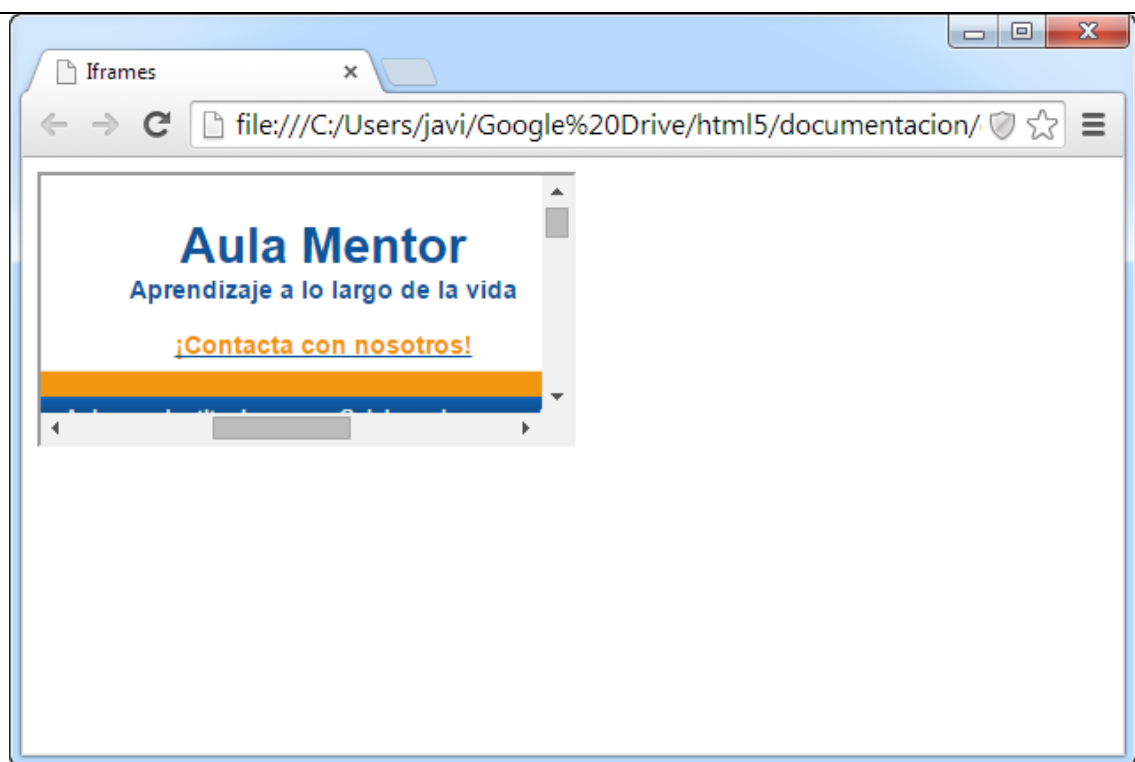
 <iframe src="http://www.mentor.mec.es">

 <p>Tu navegador no soporta iframes.</p>

 </iframe>

</body>

</html>
```



Como se ve en este ejemplo, se crea un espacio reservado para visualizar la página <http://www.mentor.mec.es> dentro de la que hemos creado. Al no poner ninguna dimensión al `iframe`, el navegador lo ha generado con ese ancho y largo. También hay que fijarse en el texto ("Tu navegador no soporta iframes") que se visualizará si el navegador no soporta el elemento `<iframe>` y que se escribe entre la etiqueta de apertura y la cierre.

En base al ejemplo anterior vemos que los `iframe` nos permiten incrustar una página web dentro de otra lo que implica que todos los elementos de esa página (enlaces, imágenes, etc.) se verán dentro del marco creado para ello.

No hay que confundir **iframe** con **frame** o marco que se empleaban en HTML4 (en HTML5 no se usan) ya que, aunque pueda parecer que hacen lo mismo, `iframe` es un elemento normal que no necesita una estructura especial como había que hacer con los `frame`.

## 2.12. Iframes



Permite visualizar un documento HTML dentro de un área de otro documento HTML.

Etiqueta	<iframe>				
Descripción	Visualiza un documento HTML dentro de un área de otro documento HTML				
Elemento	Inline				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios	name, sandbox, seamless, src, srcdoc, width y height				
Diferencia entre html 4.01 y html5	Los atributos align, frameborder, longdesc, marginheight, marginwidth y scrolling no están soportados en HTML5 pero es de los pocos elementos en el que se pueden usar los atributos height y width sin tener que utilizar estilos CSS				

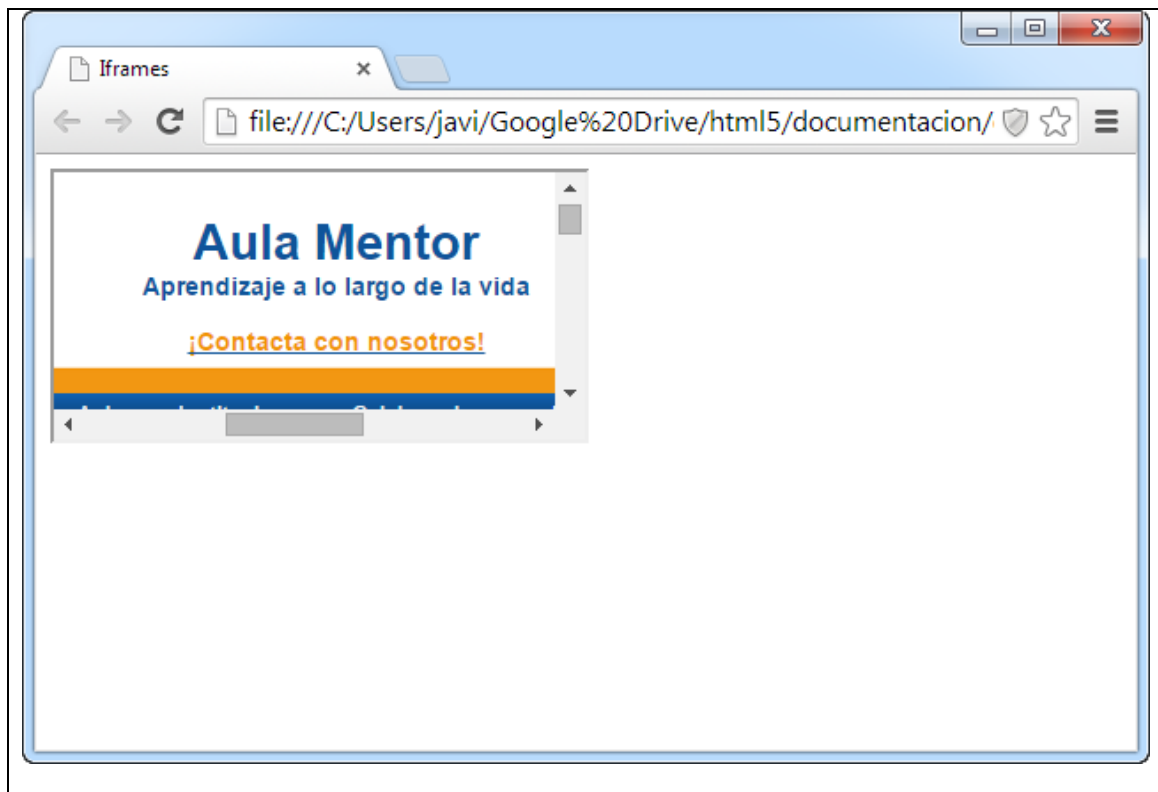
Atributos propios	Valor	Descripción
height 	pixels	Especifica la altura del elemento
name 	texto	Especifica el nombre de un marco
sandbox 	"" Allow-forms Allow-same-origin Allow-scripts Allow-top-navigation	Indica un conjunto de restricciones extras para el contenido del marco
seamless 	seamless	Especifica si el contenido del marco debe parecer como una parte de la página contenedora sin bordes ni barras de scroll

src 	URL	Especifica la URL de la página a visualizar en el marco
srcdoc 	Código html	Especifica código html para ser visualizado dentro del marco
width 	pixels	Especifica el ancho del marco

 → Nuevo en HTML5

### Ejemplo básico:

```
<!DOCTYPE html>
<html>
 <head>
 <title>Iframes</title>
 </head>
 <body>
 <iframe src="http://www.mentor.mec.es">
 <p>Tu navegador no soporta iframes.</p>
 </iframe>
 </body>
</html>
```



Los frames (frame en inglés significa marco) son unas herramientas que han tenido una historia dilatada en el desarrollo de páginas web con HTML. De ser una etiqueta no estándar ha pasado a ser soportada por todos los navegadores y formar parte de las especificaciones de HTML. Por otra parte, el frame siempre ha sido una utilidad para hacer páginas web de uso, cuando menos, controvertido, puesto que tiene ciertas desventajas que muchas veces son más importantes que la indudable practicidad.

En cualquier caso, en DesarrolloWeb.com ya hemos hablado suficiente sobre la etiqueta Frame y hemos tratado ampliamente sus ventajas e inconvenientes. En este artículo vamos a hablar de una etiqueta "hermana" que es a día de hoy mucho más usada, porque resulta más útil y menos problemática que los propios frames. Se trata de `iframe`, un tag incluido en las especificaciones de HTML 4.0.

**Nota:** `iframe` viene de inline frame, pero en castellano a veces se les llama frames flotantes

Referencias sobre la etiqueta frame:

Frames en HTML

Ventajas e inconvenientes del uso de frames

Frames - Explicación básica

Control de frames en Javascript

Frames sin bordes

Frames - Atributos avanzados

Actualizar dos frames con un solo enlace

Crear variables o funciones globales a todos los frames

Frames - Una página en cada marco

...

En concreto iframe sirve para crear un espacio dentro de la página donde se puede incrustar otra web. Es un cuadrado cuyas dimensiones debe especificar el desarrollador en la propia etiqueta iframe, que tiene asociada una página web que se carga en dicho espacio. Esa página web tendrá sus propios contenidos y estilos, independientes del contexto donde se está mostrando. Además será perfectamente funcional: si tiene enlaces se mostrarán en ese mismo espacio y si tiene scripts o aplicaciones dentro se ejecutarán también de manera autónoma en el espacio reservado al iframe.

## EJEMPLOS DE USO DE IFRAME

Iframe se utiliza en muchos contextos. Dentro de un iframe podemos mostrar contenidos de otras páginas, como si estuvieran en la nuestra, por lo que sirven para ejemplos como:

- Códigos de banner, que se invocan por medio de un iframe pidiendo los datos del banner generalmente a un servidor de banners que puede estar en otra red.
- Visualizar contenidos de terceros, como bloques de noticias o novedades que ofrecen en otras webs.
- Interfaces de usuario, en el que ciertas actividades se realizan de forma autónoma y el procesamiento está en otra página web.

## USO DE IFRAME FRENTE A FRAME

Actualmente la etiqueta iframe se utiliza más a menudo que la etiqueta frame, porque no da tantos problemas como esta. La diferencia principal está basada en que la etiqueta iframe no necesita una declaración de los espacios de los frames o frameset, porque se incrusta en el código HTML de la página. El iframe, por tanto, no provoca problemas de navegación, como los que ocurren con los frames normales si no se entra correctamente a través del frameset.

También, ya que no existe el frameset en los iframe, no adolece de los problemas del uso de frames, sobretodo a la hora en que la página es indexada en los motores de búsqueda.

Por decirlo de alguna manera, trabajar con iframe o frames flotantes es tan sencillo como hacer una tabla, que se codifica dentro de la maqueta HTML, con su espacio reservado

dentro de la página. Así, la única diferencia con respecto a una tabla es que el contenido del iframe se extrae de otra página web.

## CONSTRUCCIÓN DE LA ETIQUETA IFRAME

Como decimos, el iframe se coloca directamente en el código HTML, en el lugar donde queremos que aparezca.

Se colocaría con un código como este:

```
<iframe src="pagina_fuente.html" width=290 height=250>Texto para cuando
el navegador no conoce la etiqueta iframe</iframe>
```

Como se ve, los atributos principales de iframe son la página web que se va a mostrar en el espacio y el ancho y alto del recuadro que reservemos para el frame flotante.

Como se puede ver, la etiqueta iframe tiene su correspondiente etiqueta de cierre. Todo el texto que coloquemos entre la etiqueta de inicio y la de cierre es texto alternativo, que sólo se mostrará en caso que el navegador del visitante no acepte la etiqueta iframe.

## TODOS LOS ATRIBUTOS DE IFRAME

Estos serían los atributos disponibles para la etiqueta iframe:

**src:** Para indicar la página web que se mostrará en el espacio del frame flotante.

**width:** Para definir la anchura del recuadro del iframe.

**height:** Para definir la altura del iframe.

**name:** Para especificar el nombre del frame, que podemos utilizar luego para referirnos a él con el target de los links, o mediante javascript.

**id:** Para indicar el identificador del iframe, y poder referirnos a él desde javascript.

**frameborder:** para definir si queremos o no que haya un borde en el frame. Los valores posibles son 0 | 1. frameborder=0 indicaría que no queremos borde y frameborder=1 que sí.

**scrolling:** indica si se quiere que aparezcan barras de desplazamiento para ver los contenidos del iframe completo, en el caso que no aparezcan en el espacio reservado para el iframe. Los valores posibles son: yes | no | auto. El valor "yes" es para que aparezcan siempre las barras de desplazamiento o barras de scroll, "no" sirve para que no aparezcan nunca y "auto" es para que aparezcan sólo cuando son necesarias (es el valor por defecto).

**marginwidth:** Para definir el margen a izquierda y derecha que debe tener la página que va dentro del iframe, con respecto al borde. Este margen va en pixels, pero prevalecerá el margen que pueda tener asignada la página web que mostremos en el frame flotante.

**marginheight:** lo mismo que marginwidth, pero en este caso para el tamaño del margen por la parte de arriba y abajo.

**margin:** para especificar alineación del frame, igual que se especifica para las imágenes.

**style y class:** los atributos para definir el aspecto del iframe por medio de hojas de estilo CSS.

Para acabar, aquí vemos otro ejemplo de iframe con unos cuantos atributos más:

```
<iframe name=miframeflotante src="colabora.htm" width=400 height=550
frameborder="0" scrolling=yes marginwidth=2 marginheight=4
align=left>Tu navegador no soporta frames!!</iframe>
```

pero aparecen nuevos problemas. Por un lado un tema de seguridad, habría que controlar que nadie pase en el argumento page una dirección de un archivo que no debería poder ver. Para esto habrá que agregar unos ifs analizando el contenido. Por otro lado, absolutamente todas las páginas de contenido que quiera mostrar deberían pasar a través de este archivo. Entonces, si una página de contenido tenía un link a "descargas.html", ahora debe apuntar a "index.php?page=descargas.html". Nuevamente google me dió una buena solución. Resulta que php también tiene una par de funciones para que la salida del script vaya temporalmente a un buffer en lugar de ir directamente al navegador. Entonces, enviamos la salida del include al buffer, buscamos los links en el buffer, todo lo que empieza con href=", y los actualizamos reemplazando ese comienzo por href="index.php?page=. El result.

## POR QUÉ PASAMOS DE MARCOS

Aunque por el título pueda parecer lo contrario, no tengo nada contra nadie que se llame Marcos. Hablamos, naturalmente, de los marcos de HTML, los famosos *frames*, uno de los más odiados y más amados inventos de Netscape.

Se sabía. Se veía venir. Se lo dijeron. Los marcos eran una mala idea. Pero aun así, los implementaron.

¿Y por qué es tan mala idea?, te estarás preguntando. He aquí el porqué.

## LOS MARCOS VAN EN CONTRA DE LA PROPIA ESENCIA DEL HTML.

Es como si inventaras el tapón para mangueras.

El objetivo del lenguaje HTML es dotar a un texto de *estructura lógica*: párrafos, listas, citas, etc. En HTML, no dices **cómo** se representa cada cosa; únicamente **qué** es cada cosa. El cómo se representa depende del navegador que estés usando. En un navegador gráfico,

los títulos saldrán más grandes, los párrafos separados entre sí, saldrán marcadores en las listas, etc. Un sintetizador de voz hará pausas, inflexiones en la voz, utilizará sonidos auxiliares, etc. Un motor de búsqueda lo ignorará todo y se dedicará simplemente a indexar las palabras. Etc., etc.

Esto es el punto fuerte de HTML: es flexible, cualquier documento se adapta a cualquier modo de navegar. Cualquier aplicación puede obtener el documento y representarlo de la mejor manera posible.

Los marcos rompen con ese esquema y lo vuelven inservible. Los marcos no son información lógica, sino de presentación. No dice **qué** es cada cosa, sino **dónde** está cada cosa. Esto, obviamente, sólo tiene sentido para navegadores gráficos. Para todas las demás aplicaciones, en las que el concepto de "dónde" no tiene ningún significado (motores de búsqueda, sintetizadores de voz, etc.), el documento puede fácilmente tornarse inaccesible, o difícilmente accesible en el mejor de los casos.

En cualquier caso, introducen una componente de presentación en un lenguaje cuya misión es describir estructuras lógicas. Si lo que queremos es especificar el modo en que se representan nuestros documentos, debemos usar hojas de estilo, que son el complemento del HTML y la herramienta apropiada para ello.

Pero si esto no es razón suficiente para ti (lo cual me hace sospechar que no tienes muy claro por qué estás haciendo páginas para la Web), aquí tienes una ristra más de buenos motivos para evitar el uso de marcos.

## NO SE PUEDE ENLAZAR A UNA COMBINACIÓN PARTICULAR DE MARCOS

Una de las primeras cosas que se aprende del diseño web, es que cada recurso tiene un "nombre propio" que lo identifica y lo distingue de todos los demás. Esto muy importante, porque la esencia de la Web es que todos los recursos que forman parte de ella se encuentran enlazados entre sí (y de ahí su nombre). Naturalmente, para que un recurso sea referenciado por otro, necesita algo que lo identifique. En la red, ese identificador es un *URI* (identificador uniforme de recursos).

Las páginas web suelen identificarse con un tipo especial de URI llamado *URL* (localizador uniforme de recursos), que especifica dónde está localizado dicho recurso. Es la cadena que aparece en la barra de direcciones (por ejemplo, el URL de este documento es <http://html.conclase.net/articulos/problemasmarcos>).

Pues bien, resulta que este esquema se rompe con las páginas con marcos.

Una página con marcos está compuesta de dos grupos de elementos: por un lado el documento HTML que contiene la *definición* de los marcos (el elemento FRAMESET) junto con el estado inicial de la página, y por otro lado los documentos HTML que constituyen el *contenido* de cada marco, y que se almacenan por separado. El URL de la página con marcos es siempre el del documento que contiene el FRAMESET. Si comenzamos a navegar por el sitio, se van cargando páginas nuevas en cada marco, pero el URL es siempre el mismo. Si cualquier persona escribe ese URL en la barra de direcciones de su navegador, siempre llegará al estado inicial de la página web.

Eso quiere decir que si queremos hacer un enlace a una página particular... no podemos. Es decir, los marcos están impidiendo en gran medida el objetivo primario de la Web: enlazar unos documentos con otros. A lo sumo, lo más que podemos hacer es enlazar con el FRAMESET y dar instrucciones para avanzar desde allí.

Naturalmente, podríamos enlazar directamente con el documento HTML que nos interesa, pero como hemos quitado deliberadamente los elementos de navegación para ponerlos en un marco separado, llegaremos a una página sin salida, sin herramientas de navegación.

## DAN PROBLEMAS CON LOS BUSCADORES

Actualmente casi todos los motores de búsqueda conocen la sintaxis de los marcos y son capaces de seguir e indexar todos los documentos enlazados desde los documentos que forman el estado inicial de la página. Sin embargo, imagina esta situación: alguien busca algo en un buscador y da con uno de tus documentos. No con el FRAMESET, sino con uno de los documentos que, en teoría, debería ir en un marco. El usuario pincha, llega al documento y... no hay marcos, no hay elementos de navegación.

## SÓLO SE PUEDE CAMBIAR EL CONTENIDO DE UN MARCO AL MISMO TIEMPO

Imagina esta situación. Tenemos tres marcos. En el primero tenemos un índice general de capítulos. En el segundo tenemos un subíndice de secciones, y en el tercero los contenidos de la sección que queremos presentar. Al elegir un capítulo del índice de capítulos, queremos que cambien los contenidos del índice de secciones de ese capítulo, y que se muestren los contenidos correspondientes a la primera sección del capítulo elegido.

Pero eso no es posible, porque al elegir uno de los capítulos del índice general, sólo podemos cambiar el contenido de uno de los otros dos marcos.  
Añaden más complejidad de la necesaria

Esto es especialmente cierto en páginas con más de dos marcos, sobre todo si queremos que cambien a la vez los contenidos de más de uno de ellos con los métodos de los que hablamos más adelante.

Hay que tener en cuenta que se tarda más en cargar dos documentos pequeños por separado que uno solo más grande que contuviera la suma de los dos (esto es así debido a la naturaleza del protocolo de red utilizado en Internet). Esto es especialmente sensible en lugares con conexiones muy lentas.

## MÁS PROBLEMILLAS

- El título deja de ser informativo. El título del documento es siempre el del documento que contiene el FRAMESET, y no el del documento individual que estamos leyendo.
- En general no es posible almacenar una combinación específica de marcos en la lista de favoritos. Algunos navegadores sí pueden, pero otros no.
- Los identificadores de fragmento dejan de ser útiles, porque no se puede apuntar hacia ellos, ya que si lo hacemos nos volvemos a quedar sin barra de navegación.
- En cuanto hay más de dos marcos, la navegación por la historia del navegador (con los botones *Atrás* y *Adelante*) se hace bastante confusa (por no decir inservible).



Y las soluciones no son precisamente buenas.

Más que soluciones son trucos, y a veces crean más problemas de los que resuelven.

El problema de cambiar los contenidos de más de un marco a la vez se suele resolver anidando FRAMESETs. En el ejemplo de arriba, en vez de definir tres marcos definiríamos sólo dos. En el primero tendríamos el índice de capítulos y en el segundo cargaríamos un nuevo frameset con dos marcos, en uno de ellos cargaríamos el índice de secciones y en el segundo presentaríamos cada sección. Esto tiene la ventaja de que se puede apuntar a un capítulo en particular, aunque nos quedaríamos sin el índice de capítulos, y seguiríamos sin poder apuntar a una sección en particular o a un fragmento de ella. Además sigue sin resolver el problema de los motores de búsqueda y hará que la navegación por el sitio sea más lenta, ya que cada vez que cambiemos de capítulo habrá que cargar tres nuevos documentos en lugar de sólo dos.

Este método se puede extender al límite, y hacer que cada documento sea en realidad un FRAMESET, de modo que cada vez que queramos abrir un documento, la ventana se vacía (usando `target="_top"`) y se vuelven a dibujar todos los marcos y a rellenarlos con los documentos apropiados. Ahora podemos cambiar todos los marcos que queramos al mismo tiempo, y tenemos más libertad para apuntar a diferentes combinaciones de marcos, pero a cambio la navegación se hace mucho más lenta, ya que cada vez que seguimos un vínculo hay que volver a cargar un FRAMESET y uno o normalmente más documentos en cada marco. Además persiste el problema de los motores de búsqueda y el de los identificadores de fragmento. Y además uno de los motivos por el que mucha gente decide usar marcos es que un documento queda fijo en la pantalla sin necesidad de volver a cargarlo cada vez, cosa que con este método ya no ocurre.

Por último, existen scripts de JavaScript para resolver algunos problemas (por ejemplo, cargar automáticamente el FRAMESET cuando se accede directamente a uno de los documentos desde un motor de búsqueda). Sin embargo, uno de los principios del diseño web es que cualquier página debería funcionar aunque el usuario no pueda o no quiera activar JavaScript en su navegador. Por tanto, estos métodos constituyen una solución parcial, no una solución de ningún problema.

### **En consecuencia:**

A primera vista, los marcos son una buena idea. En el fondo, no dan más que problemas y no son tan necesarios como te podrías imaginar. Mi consejo es que, siempre que puedas, los evites.

Si de todos modos vas a usarlos o tienes que hacerlo, hazte un favor: no te unas al club de los 500.000 maleducados. Está muy bien que utilices el elemento NOFRAMES para los navegadores que no soportan marcos, pero es absurdo utilizarlo para decir Su navegador no acepta marcos. Y es una auténtica falta de respeto decir Actualice su navegador, o Quizá sea un buen momento para modernizarse cuando no sabes si la persona que está leyendo tu mensaje puede realmente actualizar su navegador o no. Incluso podría estar usando un lector de pantalla carísimo para utilizar con un navegador de texto. La verdad es que Lo siento, no soporto su navegador está más cercano a la realidad. Pero en todo caso, si vas a poner algo en el elemento NOFRAMES, que sea información. Recuerda que todo esto va de transmitir información. Pon un vínculo a una página, o a varias. Sé amable con todos tus visitantes y no despidas a algunos de ellos con un mensaje grosero sólo porque no navegan como tú. Y piensa en ir diseñando una nueva versión de tu página... sin marcos.

### **Más información:**

Este artículo se basa en el tutorial sobre marcos de Stephanos Piperoglou en [webreference.com](http://webreference.com). En [allmyfaqs.com](http://allmyfaqs.com) puedes encontrar muchos más documentos con

información sobre los problemas de los marcos, algunos más modernos que otros, pero todos ellos con información interesante.

© 2001-2003 Juan R. Pozo

## **BLOQUE 3:**

# **LOS NUEVOS ELEMENTOS SEMÁNTICOS HTML5**

### 3. LOS NUEVOS ELEMENTOS SEMÁNTICOS HTML5.

En este tema vamos a conocer las innovaciones que incorpora HTML5 en forma de elementos semánticos tanto para la estructura como para el aspecto de nuestra página web.

Hay que recordar que siempre que hacemos un documento HTML, éste tiene muchas similitudes estructurales con bastantes páginas HTML, es decir, que básicamente llevan encabezados, bloques de textos, pies, etc., por lo que HTML5 incorpora una serie de elementos semánticos, elementos que aportan información relevante de cómo es su contenido, que suplirán al elemento `<div>` a la hora de hacer bloques en el documento ya que `<div>` no es un elemento semántico, es decir, no sabemos nada del tipo de contenido que lleva dentro.

Por ahora todos estos elementos semánticos (`<header>`, `<section>`, etc.) funcionan igual que el elemento `<div>` pero en un futuro próximo los navegadores deberían diferenciarlos y tratarlos de diferente manera ya que el contenido de un `<header>` o cabecera no puede tener el mismo tratamiento que el contenido de una sección o `<section>`.

### 3.1. ESTRUCTURA DE UNA PÁGINA HTML5

Tal y como hemos visto, la estructura general de cualquier página HTML5 es la siguiente:

```
<!DOCTYPE html>
<html>
 <head>
 .
 .
 </head>
 <body>
 .
 .
 </body>
</html>
```

Esta estructura se podría generalizar a cualquier versión de HTML e, incluso de XHTML cambiando el contenido de la declaración `<!DOCTYPE>`

También hemos visto que se pueden crear divisiones o partes dentro de `<body>`, para separar contenidos con el elemento `<div>`, lo cual también es general de cualquier versión HTML o XHTML, pero `<div>` tiene el gran inconveniente de que no sabemos qué tipo de contenido hay dentro de él.

Una de las principales innovaciones que ofrece HTML5 es que el cuerpo de las páginas, o sea, lo que se encuentra dentro de `<body>` y `</body>`, se organiza en apartados o secciones que dan un sentido más coherente a su estructura (de hecho `<body>` se podría considerar como una sección dentro de la cual se crean todas las demás). Esto trae como consecuencia que la utilización de `<div>` se reduzca drásticamente hasta casi estar en desuso.



*Usa los nuevos elementos semánticos HTML5 en vez de `<div>` siempre que puedas*

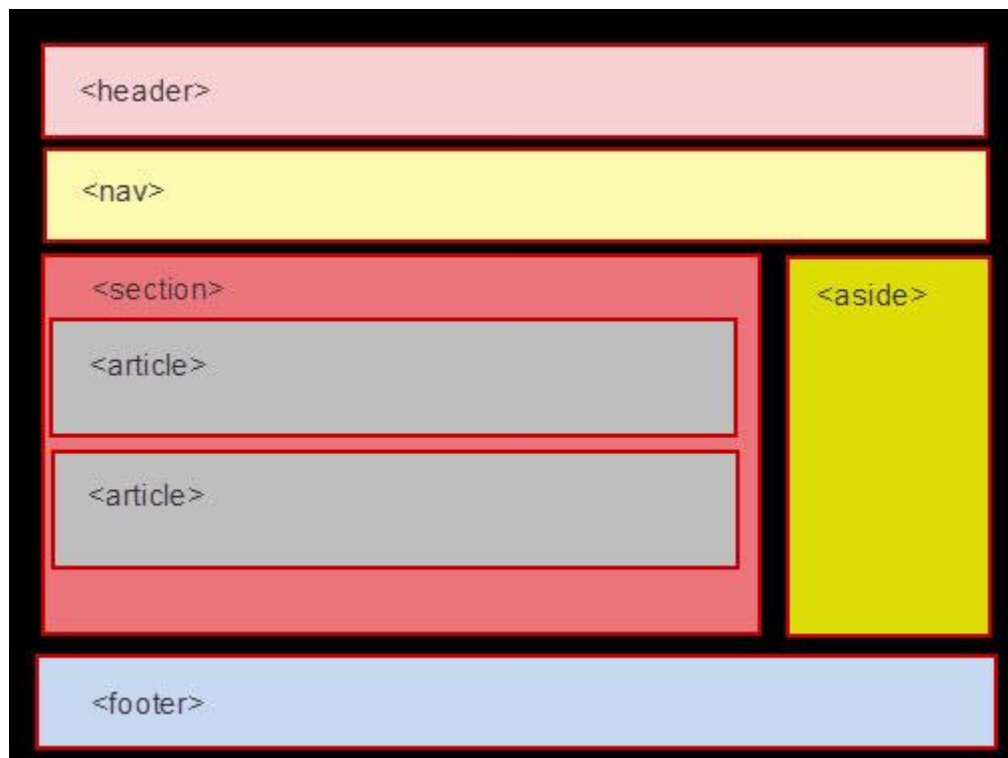
Todas estas partes se definen a través de las nuevas etiquetas semánticas exclusivas de HTML5. Las etiquetas semánticas, a diferencia de las etiquetas normales, se fijan más en indicar el tipo de contenido que tienen en vez de cómo se formatea su información.

Las anteriores etiquetas forman parte de los elementos semánticos que van a ser los que realmente formen la estructura del documento HTML5. Estos elementos no tienen un lugar fijo en el cuerpo de la página lo cual da bastante flexibilidad a dicha estructura.



*A lo largo del curso hemos manejado mucho el elemento `<div>` pero todo lo visto sobre él se puede extrapolar al uso de los nuevos elementos semánticos HTML5.*

Un ejemplo gráfico de una posible estructura del cuerpo o `<body>` con los nuevos elementos de una página HTML5 podría ser:



### 3.2. ELEMENTOS SEMÁNTICOS HTML5

Un elemento semántico describe claramente su significado tanto al navegador como a la persona que lee el código del documento, es decir, indica el tipo de contenido que está entre su etiqueta de apertura y su etiqueta de cierre.

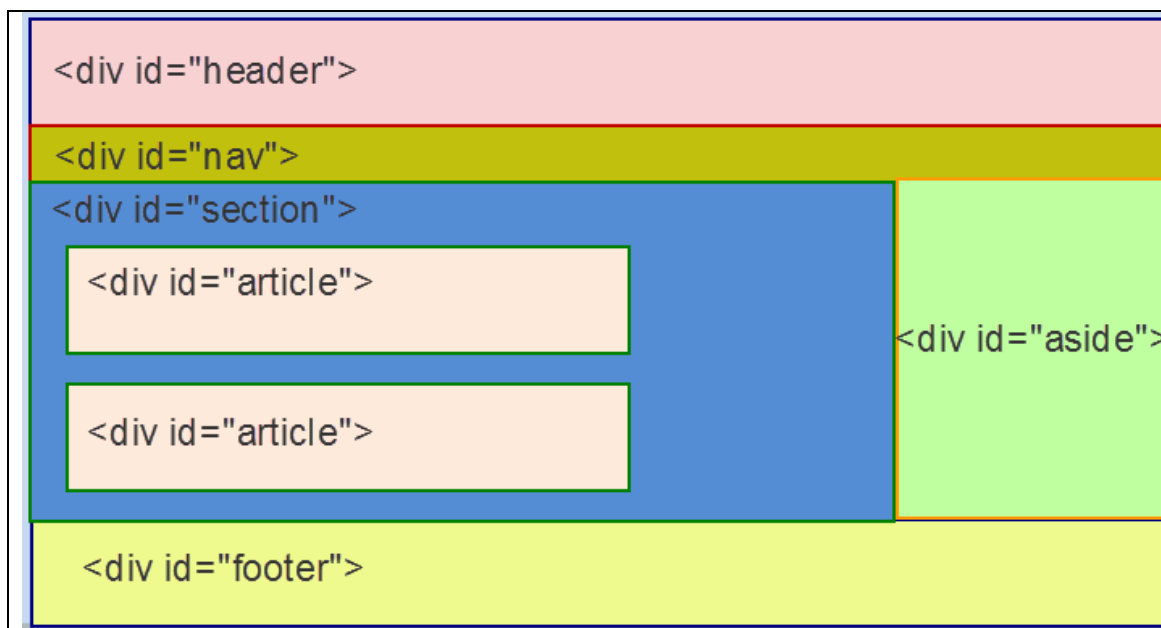
Ejemplos de elementos semánticos: `<table>`, `<img>`, `<section>`, etc.

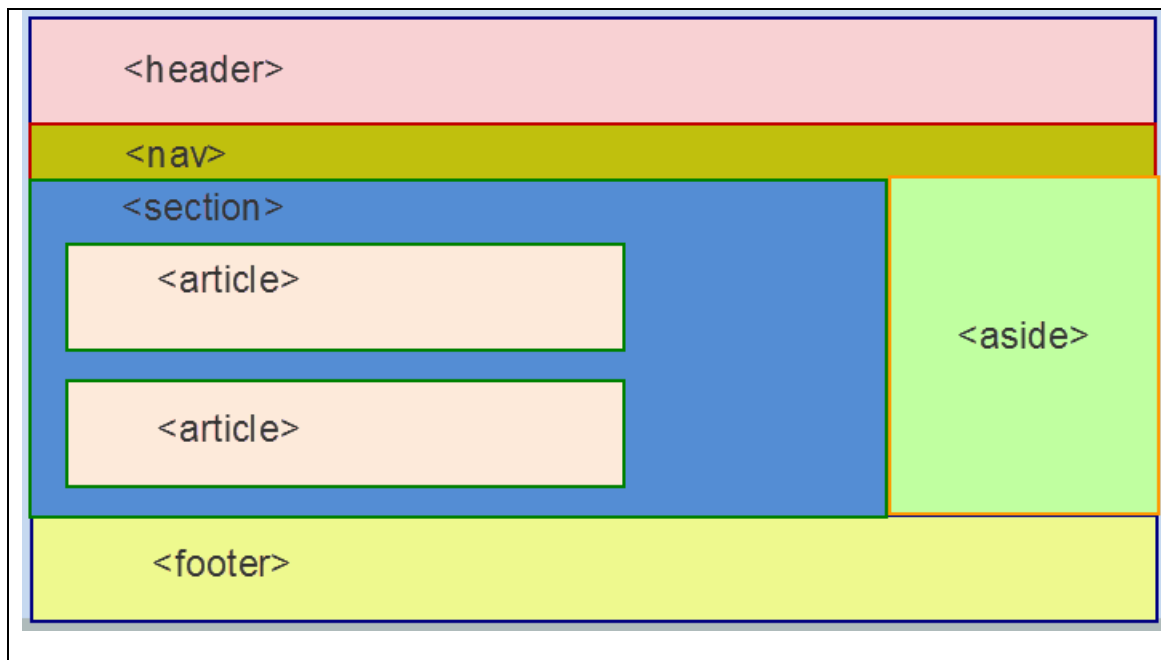
De la definición anterior se desprende que un elemento no semántico es aquel no da ninguna información sobre la naturaleza del contenido que está entre su etiqueta de apertura y su etiqueta de cierre.

Ejemplos de elementos no semánticos: `<div>`, `<span>`, etc.

	<i>Los elementos semánticos no son algo nuevo de HTML5 ya que las anteriores versiones de HTML los utilizan. Los que sin son nuevos son los elementos semánticos <code>&lt;section&gt;</code>, <code>&lt;article&gt;</code>, etc., con los que se intenta crear una estructura más coherente de una página HTML5.</i>
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

En la versión anterior a HTML5, o sea, HTML4, los nuevos elementos semánticos de estructura se podían representar a través elementos `<div>` a los cuales se les asignaba una identificación (parámetro **id**) que permitía diferenciarlos entre si y aplicarles un determinado estilo CSS.





En la primera imagen se ve como se podría crear en HTML4 una estructura similar a la que usa el estándar HTML5 (segunda imagen) empleando elementos `<div>`.

### 3.2.1. Encabezados o títulos de secciones

Antes de ver los nuevos elementos estructurales, vamos a ver los elementos que nos van a permitir poner títulos a estas partes o secciones.



*Estos elementos pueden ser sustituidos por textos tratados con estilos CSS*

- **`<h1>` .. `<h6>`:** Son elementos que formatean textos o encabezados a un tamaño fijo y determinado.

Etiqueta	<h1> .. <h6>					
Descripción	Definen encabezados de un tamaño determinado					
Elemento	Bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					
Atributos propios	Ninguno					
Diferencia entre html 4.01 y html5	El atributo align no es soportado en HTML5					



Su principal función es resaltar de una determinada manera alguna parte específica de nuestra página como, por ejemplo, un título. Lo normal es usar estos elementos con arreglo a la importancia del texto que queremos remarcar (con <h1> para lo más importante y <h6> para lo menos importante).



*El tamaño del texto normal con el que se ve los contenidos de nuestras páginas es igual al texto formateado con el elemento <h4>*

Ejemplo comparativo de elementos <h1>, <h2>, <h3>, <h4>, <h5> y <h6>

```
<!DOCTYPE html>
<html>
<head>
 <title>Cabeceras </title>
 <meta charset="UTF-8">
</head>
<body>
 <h1> Texto muy grande con elemento h1</h1>
 <h2> Texto grande con elemento h2 </h2>
 <h3> Texto un poco mayor que el normal con elemento
 h3</h3>
 <h4> Texto de tamaño normal con elemento h4</h4>
 <h5> Texto pequeño con elemento h5 </h5>
 <h6> Texto muy pequeño con elemento h6</h6>
</body>
</html>
```



### 3.2.2. Los nuevos elementos estructurales de HTML5

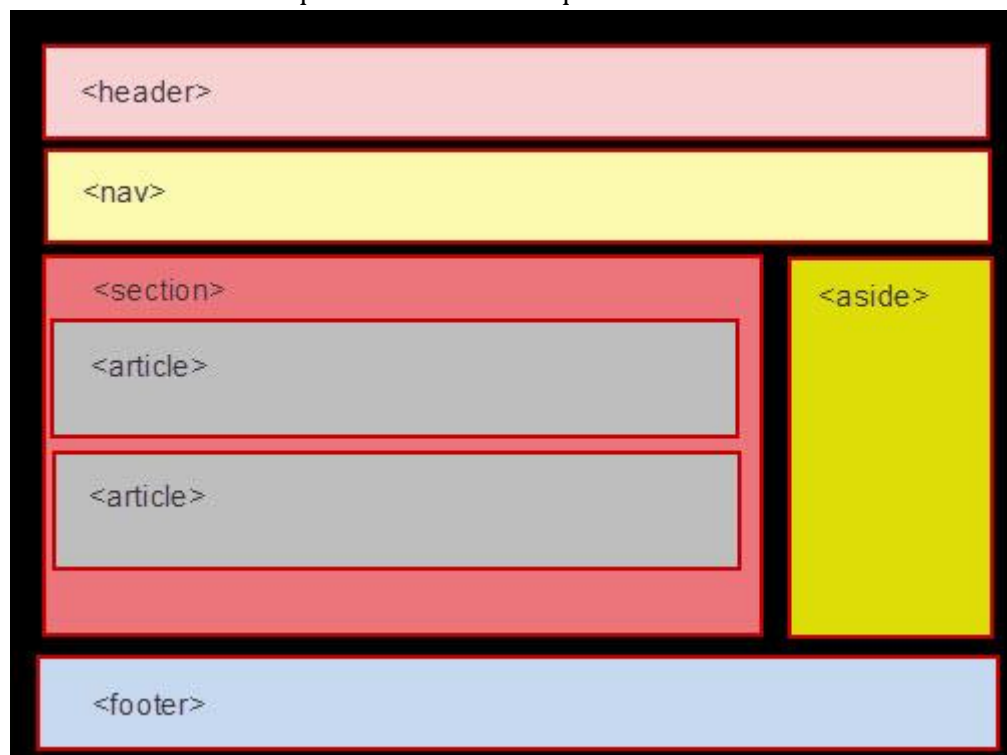
Tal y como hemos explicado en el primer punto del tema, estos elementos estructurales han sido añadidos a HTML5, esto implica que las anteriores versiones de HTML no los reconocen como elementos HTML, y que sirven para crear la estructura de una página HTML5.

Por ahora estos elementos no son obligatorios de usar y todos los navegadores los tratan como si fueran `<div>` pero, en el futuro, todos los navegadores deberán distinguirlos y conocer su función específica por lo que es muy conveniente irse adaptando a ellos e irlos empleando en nuestras páginas web en vez de `<div>`.

Las etiquetas de todos estos elementos tienen características comunes que se reflejan en la siguiente tabla:

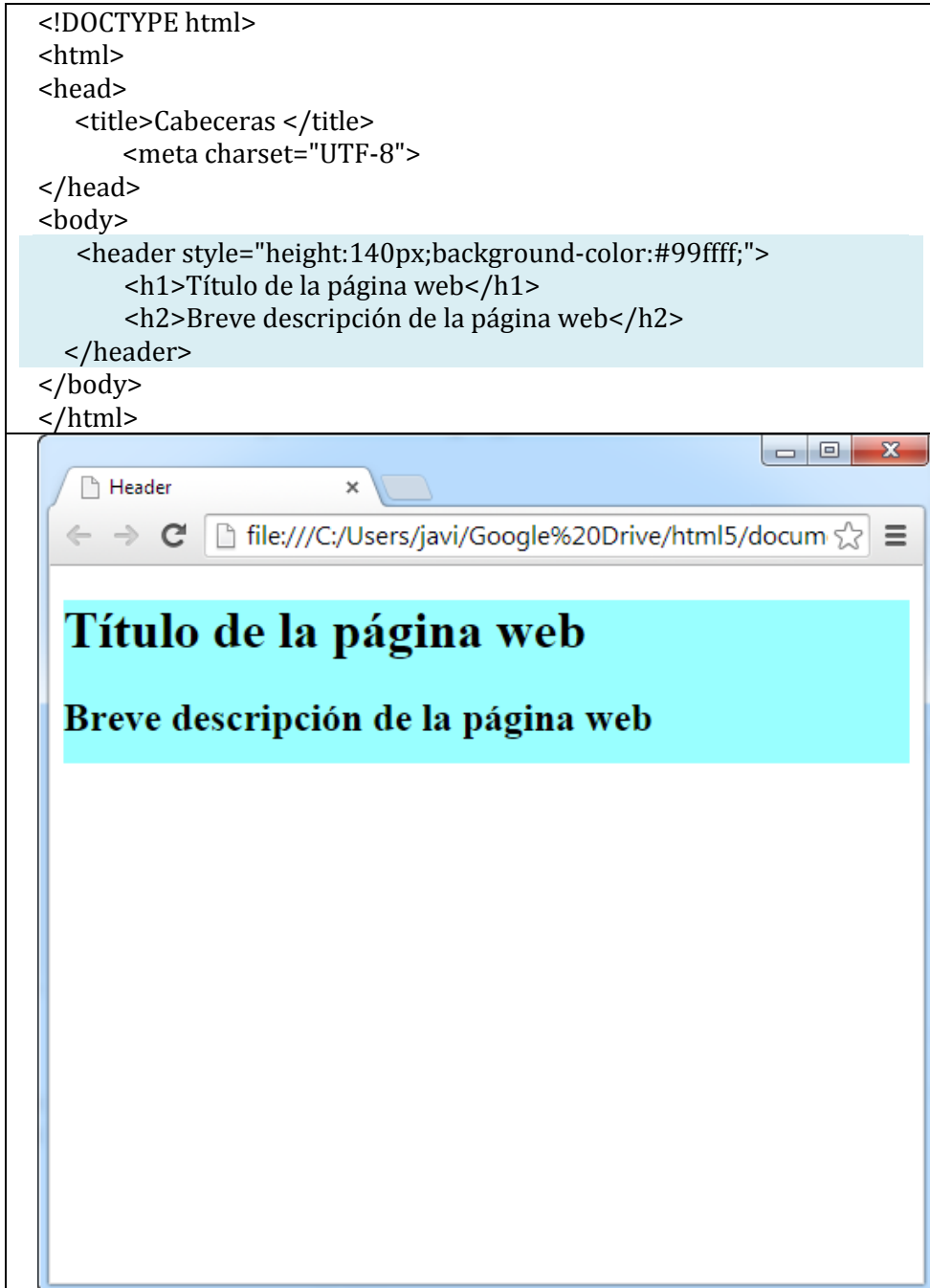
Etiqueta	<header> <nav> <section> <article> <aside> <footer>				
Elemento	Bloque				
Navegadores que las soportan y a partir de qué versión					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	6.0	4.0	9.0	11.1	5.0
Atributos	Globales y de eventos				
Atributos propios	No tienen				
Diferencia entre html 4.01 y html5	No soportadas por ninguna versión anterior a HTML5				

Fíjate en la imagen siguiente ya que es un ejemplo de una posible estructura de una página HTML5, y vamos a ir construyendo poco a poco un documento html con esa distribución a medida que vamos viendo lo que hacen cada uno de los elementos.



- **<header>**: Es el equivalente a la cabecera de un elemento o de la página web. Suele contener el título ya sea de la página o del elemento, logotipos e información relacionada.

Ejemplo de cabecera de una página web:



Como vemos en el ejemplo hemos creado un elemento `<header>` de una determinada altura (140 píxeles) y un color de fondo con un título y una breve descripción del contenido de la página.

- **<nav>**: Este apartado contiene los enlaces (barra de navegación) externos o internos de la página. Si `<nav>` no está dentro de ningún otro elemento los enlaces que lleve son, por así decirlo, los más importantes de la página ya que puede haber otros

enlaces en las diferentes partes del documento ya sea dentro de otro elemento <nav> o no.

Vamos a agregar al ejemplo anterior una sección <nav>

```
<!DOCTYPE html>
<html>
<head>
 <title>Cabecera y nav </title>
 <meta charset="UTF-8">
</head>
<body>
 <header style="height:140px;background-color:#99ffff;">
 <h1>Título de la página web</h1>
 <h2>Breve descripción de la página web</h2>
 </header>
 <nav style="height:40px;background-color:grey;">
 Proyecto
Mentor &~
 Google
 &~
 IES Virgen del Espino
 </nav>
</body>
</html>
```



En el ejemplo hemos creado una sección `<nav>` con tres enlaces a diferentes páginas separadas por espacios en blanco (`&nbsp;`).

- **`<section>`**: Es un bloque de elementos y/o textos que tienen relación entre sí. Este apartado puede agrupar diferentes subapartados que pueden ser o no de tipo `<article>`, del tema que trate.

Continuando con la construcción de nuestra página HTML5, agregamos al ejemplo anterior un apartado `<section>` con un título, un párrafo de texto, de una determinada altura y que no ocupa todo el ancho de la página.

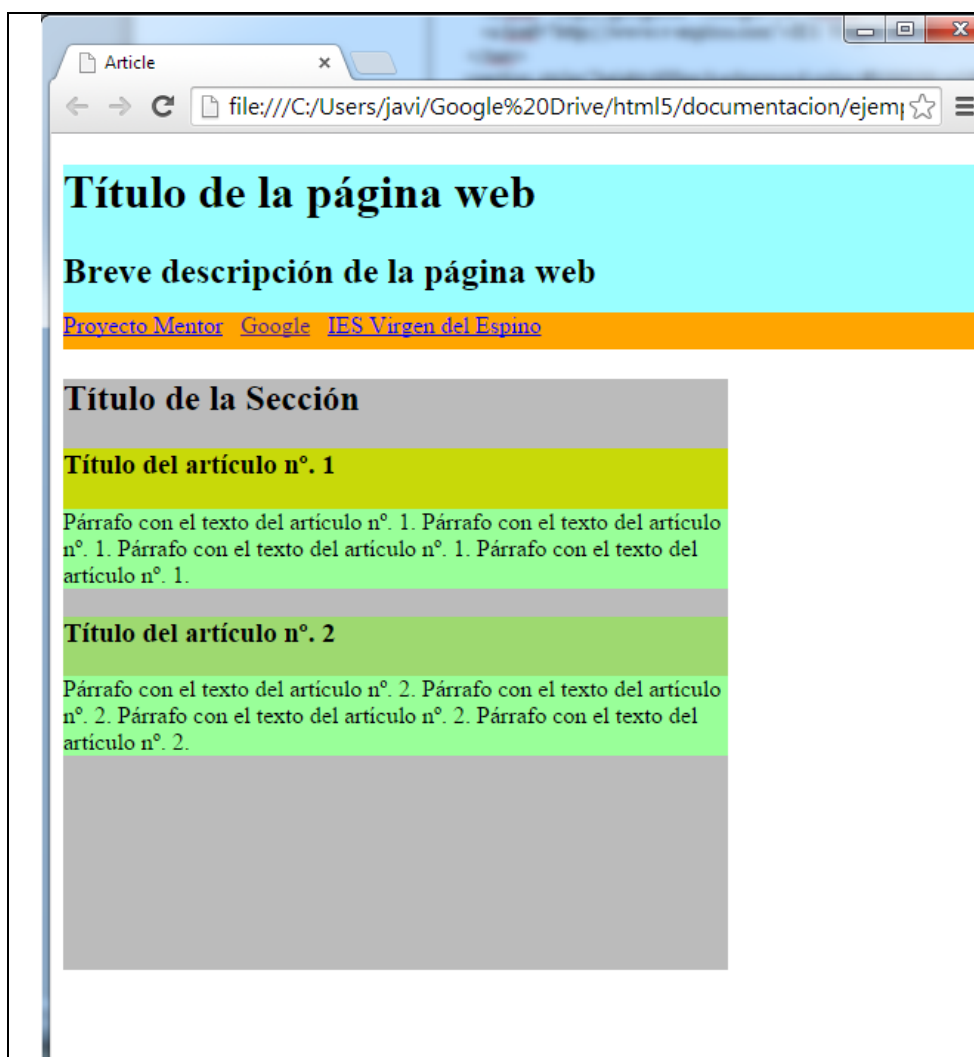
```
<!DOCTYPE html>
<html>
 <head>
 <title>Section </title>
 <meta charset="UTF-8">
 </head>
 <body>
 <header style="height:100px;background-color:#99ffff;">
 <h1>Título de la página web</h1>
 <h2>Breve descripción de la página web</h2>
 </header>
 <nav style="height:25px;background-color:orange;">
 Proyecto
```

```
Mentor

Google
IES Virgen del Espino
</nav>
<section style="height:400px;background-
color:#bbbbbb;width:450px;">
 <h1> Título de la Sección</h1>
 <p style="background-color:#99ff99;">
 Párrafo con el texto de la sección.
 Párrafo con el texto de la sección.
 Párrafo con el texto de la sección.
 Párrafo con el texto de la sección.
 Párrafo con el texto de la sección.
 Párrafo con el texto de la sección.
 Párrafo con el texto de la sección.
 Párrafo con el texto de la sección.
 </p>
</section>
</body>
</html>
```



[illegible]



- **<aside>**: Define un bloque de contenido que no es esencial dentro de la información de la página o elemento dentro del que se defina pero que puede servir para completar dicha información.

Si te fijas, hemos dejado un espacio al lado del elemento `<section>` para poner ahí el elemento `<aside>`. Ahora tenemos que solucionar un pequeño problema y es que tanto `<section>` como `<aside>` como cualquiera de los elementos que estamos viendo en este apartado son elementos bloque lo que implica que no se puede poner ningún otro elemento al lado a no ser que utilicemos propiedades de estilos CSS. Para ello vamos a usar la propiedad **display:table-cell** de CSS3 tanto en el elemento `<section>` como en el `<aside>` que hace que se pueda poner un elemento bloque junto a otro (en el siguiente bloque de estilos CSS3 del curso veremos más detenidamente esta propiedad)

```
<!DOCTYPE html>
<html>
<head>
 <title>Aside </title>
 <meta charset="UTF-8">
</head>
<body>
```



```
<aside style="height:400px;background-color:red;
 display:table-cell;width:25%;">
<h3> Título Bloque aside</h3>
<p style="background-color:#99ff99;">
 Párrafo con el texto del bloque aside.
 Párrafo con el texto del bloque aside.
 Párrafo con el texto del bloque aside.
</p>
</aside>
```



- **<footer>**: Equivale al pie de página o de cualquier otro elemento. En caso de usarse como pie de página el contenido que se suele añadir es: información de contacto, autor, copyright, etc.

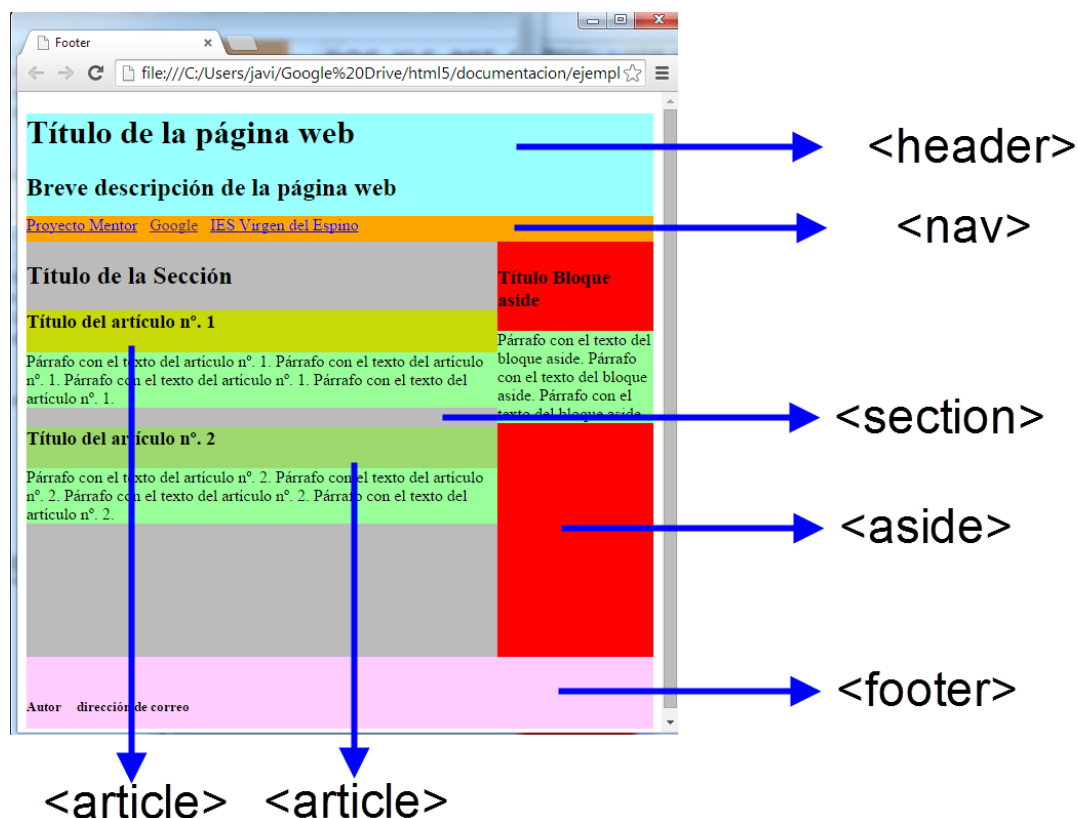
Vamos a agregar un pie de página con un ancho de 70 píxeles y con el texto “autor dirección de correo” empleando una cabecera de menor tamaño que el texto normal (<h5>).

```
<!DOCTYPE html>
<html>
<head>
 <title>Aside </title>
 <meta charset="UTF-8">
</head>
<body>
 <header style="height:100px;background-color:#99ffff;">
 <h1>Título de la página web</h1>
 <h2>Breve descripción de la página web</h2>
 </header>
 <nav style="height:25px;background-color:orange;">
 Proyecto
```





Con lo que ya tenemos un ejemplo de estructura de una página HTML5 creada con las nuevas etiquetas semánticas.



La estructura que hemos creado en el apartado anterior es totalmente orientativa, es decir, nuestras páginas no tienen por qué respetarla ya que los elementos que hemos empleado son opcionales y no tienen por qué estar en ese orden. También debemos tener en cuenta que estos elementos se pueden combinar entre sí y que pueden estar dentro unos de otros.

Un elemento `<section>` puede tener definidos dentro elementos `<header>`, `<nav>`, `<articles>`, `<aside>`, `<footer>` e, incluso, otros `<section>` al igual que un elemento `<article>` puede tener definidos dentro elementos `<header>`, `<nav>`, `<section>`, `<aside>`, `<footer>` e, incluso, otros `<article>`. En cambio, lo normal es que `<nav>` sólo tenga enlaces, `<footer>` información del autor de la página y otros datos de interés y `<aside>` contenidos relacionados con la temática de la página.

Pero ¿es igual un `<header>` cabecera principal de una página que un `<header>` cabecera de un `<section>` o de un `<article>`? Evidentemente no. Una cabecera de una página debe tener títulos más grandes, por ejemplo, que el de un `<section>` o `<article>`. Llegados a este punto vemos que en nuestras páginas vamos a usar elementos que tienen una misma función pero que no son iguales y es aquí donde entra ya de lleno el tema de los estilos CSS (cosa que abordaremos con más extensión en el siguiente bloque del curso) por lo que lo primero que vamos a hacer es identificar y diferenciar esos elementos a través del atributo global **id** (texto que sirve para esta identificación del elemento en cuestión) y así, posteriormente, poder aplicarles un determinado estilo.



*También podemos poner los mismos estilos a diferentes elementos por medio de su **id** lo que simplifica el código al no tener que repetirlo por cada elemento al que se aplique.*

La aplicación del atributo global **id** a cualquier elemento es bastante simple, ya que basta con agregar dicho atributo con su valor a la etiqueta de apertura.

```
<etiqueta id="identificador">
```

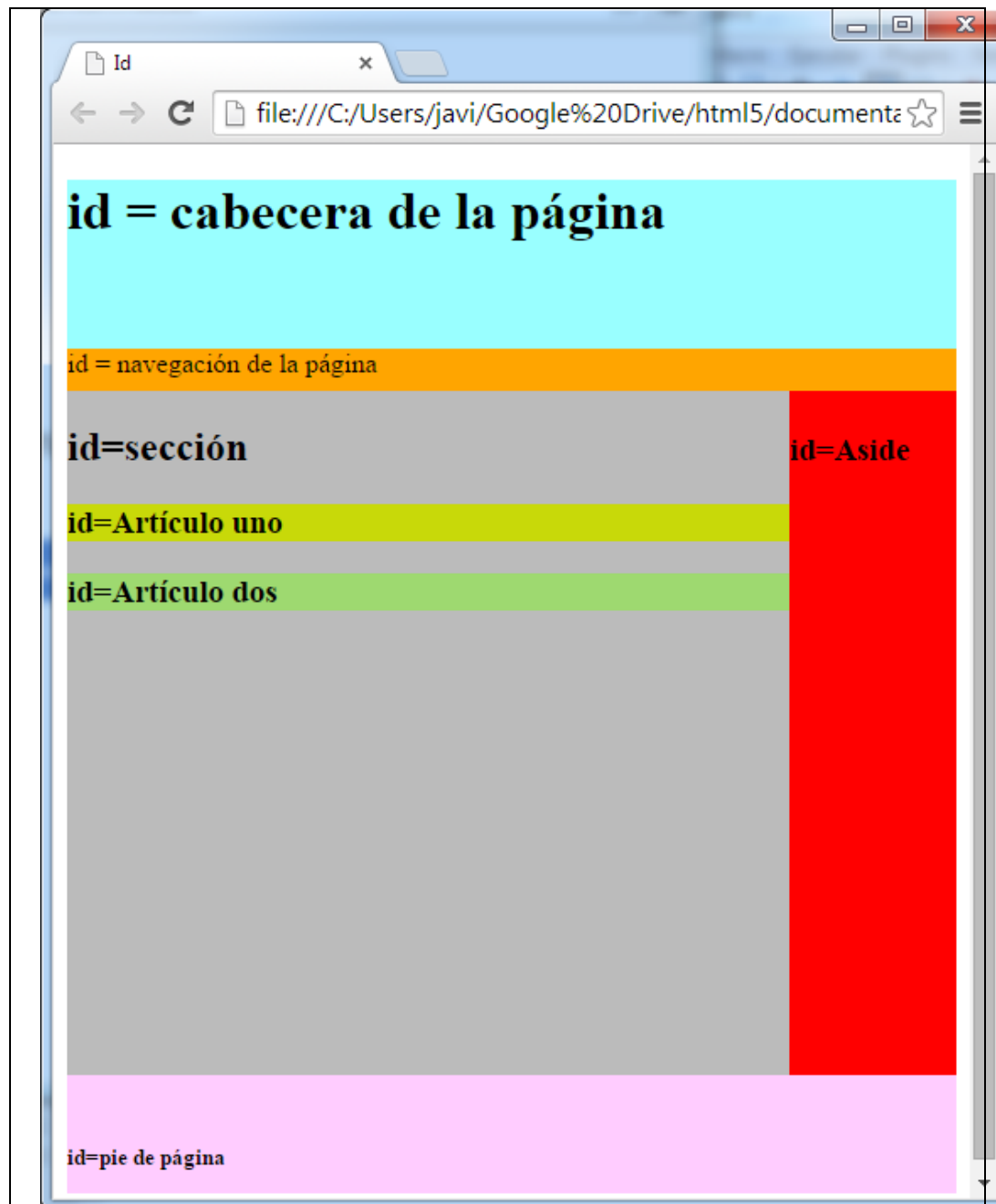
Por ejemplo:

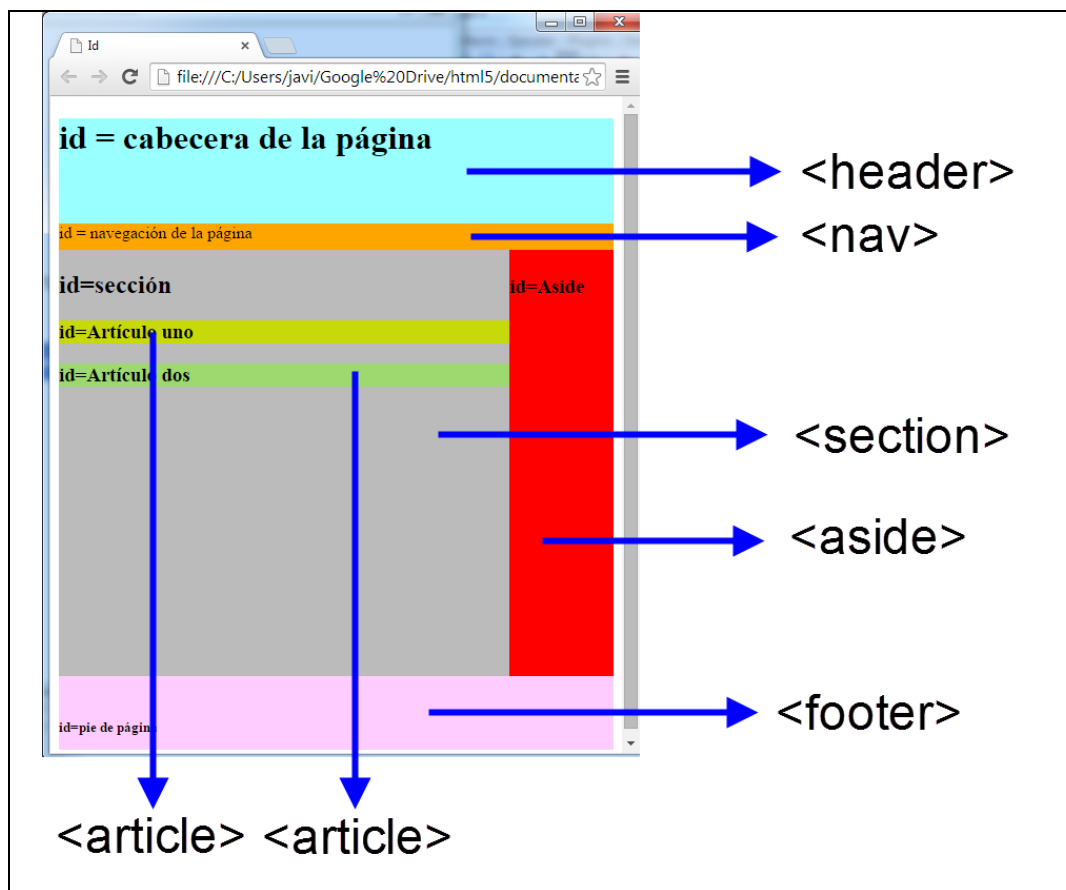
```
<header id="Cabecera de la página">
```

Vamos a poner el atributo **id** a los elementos creados en los ejemplos anteriores

```
<!DOCTYPE html>
<html>
<head>
 <title>Id</title>
 <meta charset="UTF-8">
</head>
<body>
<header id="cabecera de la página" style="height:100px;background-
color:#99ffff;">
 <h1>id = cabecera de la página</h1>
</header>
<nav id="navegación principal" style="height:25px;background-color:orange;">
 id = navegación de la página
</nav>
<section id="sección" style="height:400px;background-
color:#bbbbbb;width:450px;
display:table-cell;">
 <h2> id=sección</h2>
 <article id="Artículo uno" style="background-color:#c8d909;">
 <h3> id=Artículo uno</h3>
 </article>
 <article id="Artículo dos" style="background-color:#9ed970;">
 <h3>id=Artículo dos</h3>
 </article>
</section>
<aside id="Aside" style="height:400px;background-color:red;
display:table-cell;width:100px;">
 <h3> id=Aside</h3>
</aside>
<footer id="pie de página" style="height:70px;background-color:#ffccff;">

 <h5>id=pie de página</h5>
</footer>
</body>
</html>
```





**Ejercicio 26:** Tomando como base el ejemplo anterior vamos a trabajar con el elemento `<section>` modificándole el ancho a 600 px, con el identificador “sección” y agregándole lo siguiente:

- Utiliza los colores de fondo que creas conveniente para cada uno de los elementos.
- Una cabecera `<header>` de 40 px de ancho con el identificador “cabecera de sección” con el texto “Cabecera de la sección” aplicándole el elemento `<h3>`
- Una barra de navegación `<nav>` de 30 px de ancho con el identificador “barra de navegación de sección” con el texto “enlaces de la sección”
- Un artículo `<article>` de 250px de ancho con el identificador “artículo 1 de sección” con el texto “artículo nº. 1”
- Otro artículo `<article>` de 250px de ancho con el identificador “artículo 2 de sección” con el texto “artículo nº. 2”
- Un pie `<footer>` de 30px de ancho con el identificador “pie de sección” con el texto “Más información” aplicándole el elemento `<h5>`
- Guarda el ejercicio como ejercicio26.html
- 

### 3.2.3. Otros nuevos elementos semánticos HTML5 relacionados con el aspecto.

- **<details>**: Permite añadir información adicional que el usuario puede visualizar u ocultar.



Etiqueta	<details>				
Descripción	Añade información que se puede visualizar u ocultar				
Elemento	Bloque				
Navegadores que la soportan y a partir de qué versión					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	12.0	--	--	15.0	6.0
Atributos	Globales y eventos				
Atributos propios	open				
Diferencia entre html 4.01 y html5	No soportada por ninguna versión anterior a HTML5				

Atributos propios	Valor	Descripción
open 	open	Especifica si, por defecto, la información es visible (open) o no

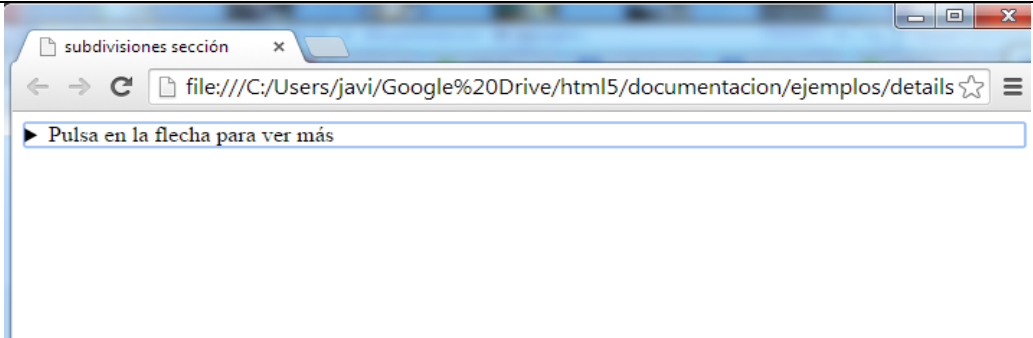
	<i>El elemento &lt;summary&gt; va dentro de &lt;details&gt; y permite visualizar un texto cuando &lt;details&gt; no está abierto o desplegado.</i>
------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------

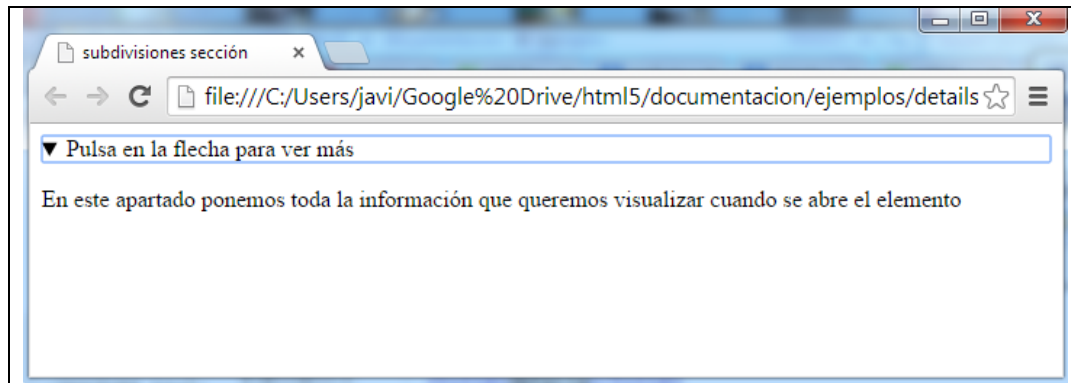
### Ejemplo:

```

<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>details</title>
 <meta charset="UTF-8">
</head>
<body>
 <details>
 <summary>Pulsa en la flecha para ver más</summary>
 <p> En este apartado ponemos toda la información
 que queremos visualizar cuando se abre el elemento</p>
 </details>
</body>
</html>

```





- **<figure>**: Elemento que contiene una imagen, gráfico, diagrama, etc., y al que se le puede poner una leyenda para informar sobre su contenido.

Etiqueta	<figure>				
Descripción	Especifica que el contenido es una imagen, gráfico, dibujo, etc.				
Elemento	Bloque				
Navegadores que la soportan y a partir de qué versión					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	6.0	4.0	9.0	11.1	5.0
Atributos	Globales y eventos				
Atributos propios	No tiene				
Diferencia entre html 4.01 y html5	No soportada por ninguna versión anterior a HTML5				

	<i>El elemento &lt;figcaption&gt; va dentro de &lt;figure&gt; y permite agregar un título o texto explicativo a éste.</i>
-------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------

### Ejemplo:

```

<!DOCTYPE html>
<html lang="es-es">
 <head>
 <title>figure</title>
 <meta charset="UTF-8">
 </head>
 <body>
 <figure>
 <figcaption> Ejemplo de título de imagen</figcaption>

 </figure>
 </body>
</html>

```



- **<hgroup>**: Agrupa encabezados <h1>, <h2>, ..., <h6>

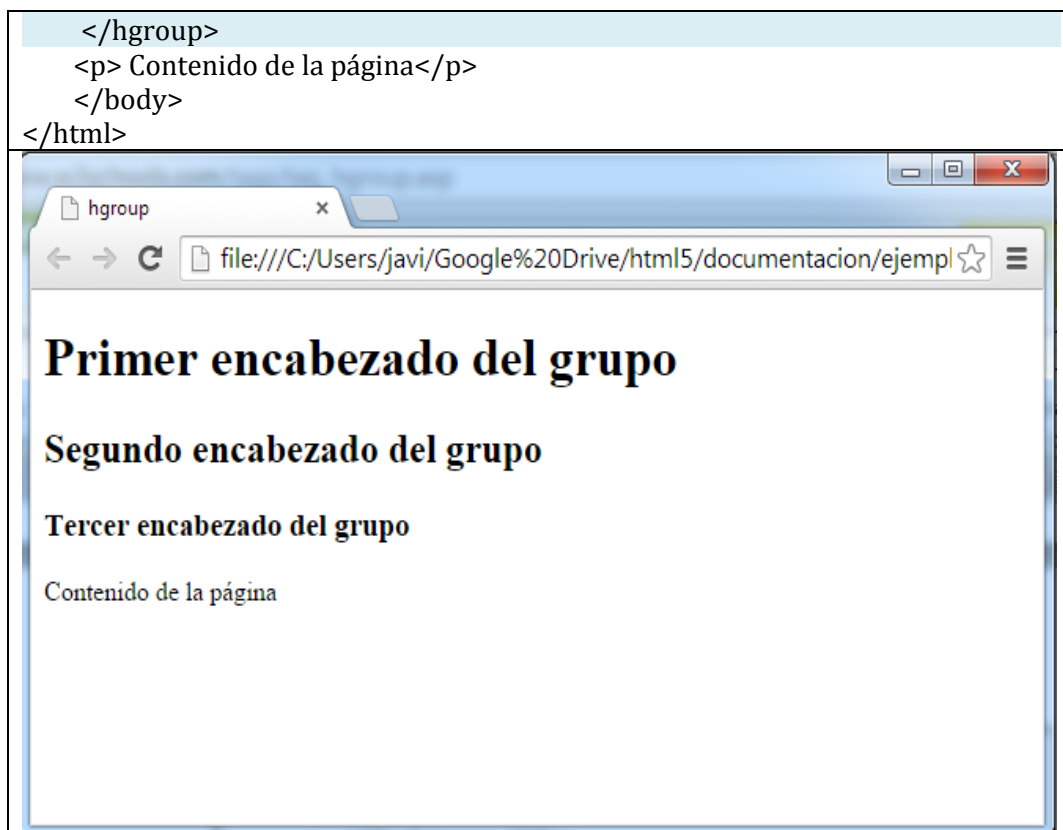
Etiqueta	<hgroup>				
Descripción	Agrupa encabezados <h1>, <h2>, ..., <h6>				
Elemento	Bloque				
Navegadores que la soportan y a partir de qué versión					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	5.0	4.0	9.0	11.1	4.1
Atributos	Globales y eventos				
Atributos propios	No tiene				
Diferencia entre html 4.01 y html5	No soportada por ninguna versión anterior a HTML5				

### Ejemplo:

```

<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>hgroup</title>
 <meta charset="UTF-8">
</head>
<body>
 <hgroup>
 <h1>Primer encabezado del grupo</h1>
 <h2>Segundo encabezado del grupo</h2>
 <h3>Tercer encabezado del grupo </h3>
 </hgroup>

```



- **<main>**: Especifica el contenido principal de una documento HTML5

Etiqueta	<main>				
Descripción	Especifica el contenido principal de un documento HTML5				
Elemento	Bloque				
Navegadores que la soportan y a partir de qué versión					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	6.0	4.0	--	11.1	5.0
Atributos	Globales y eventos				
Atributos propios	No tiene				
Diferencia entre html 4.01 y html5	No soportada por ninguna versión anterior a HTML5				

	<i>El elemento &lt;main&gt; debe ser único en el documento y puede contener elementos estructurales como &lt;article&gt;, &lt;section&gt;, etc.</i>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------

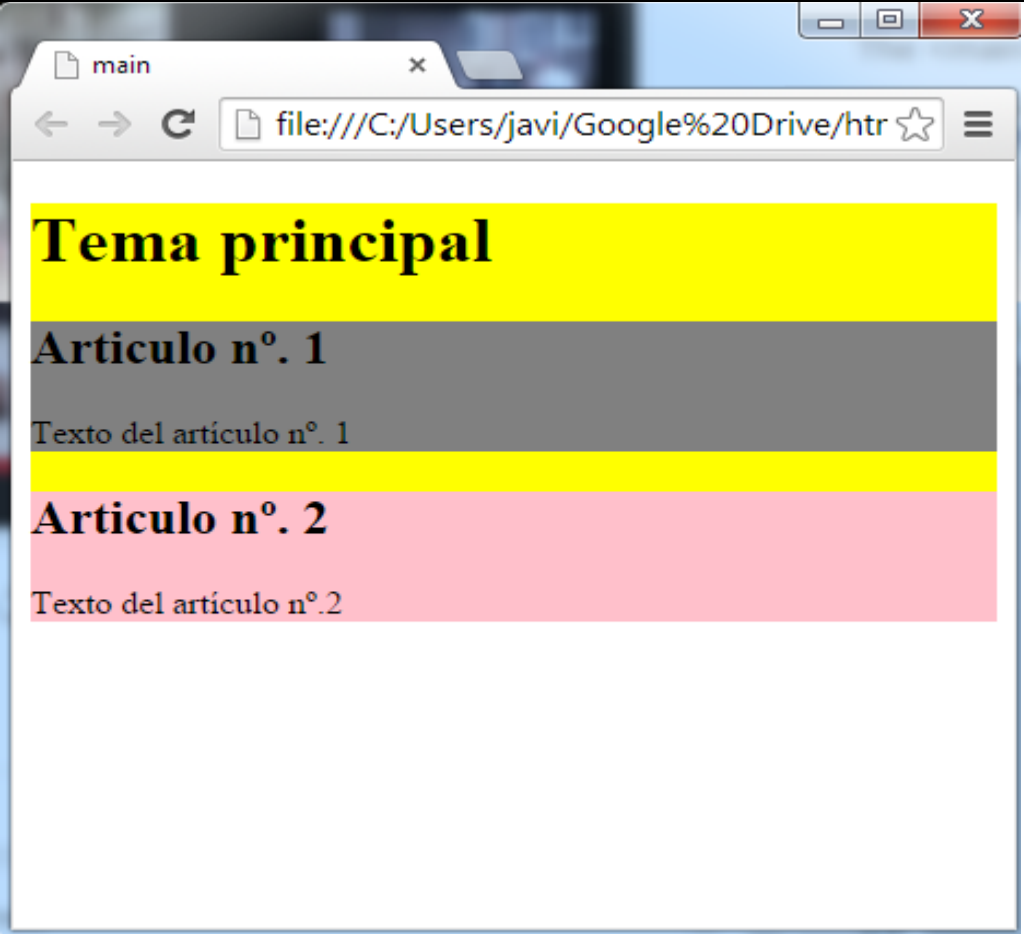
	<i>Los otros elementos semánticos utilizados para la estructura de una página HTML5 como son &lt;article&gt;, &lt;section&gt;, etc., no deben contener el elemento &lt;main&gt;.</i>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**Ejemplo:**

```

<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>main</title>
 <meta charset="UTF-8">
</head>
<body>
 <main style="background-color:yellow;">
 <h1>Tema principal</h1>
 <article style="background-color:grey;">
 <h2>Articulo nº. 1</h2>
 Texto del artículo nº. 1
 </article>
 <article style="background-color:pink;">
 <h2>Articulo nº. 2</h2>
 Texto del artículo nº.2
 </article>
 </main>
</body>
</html>

```



- **<mark>**: Marca, normalmente, una parte de un texto.

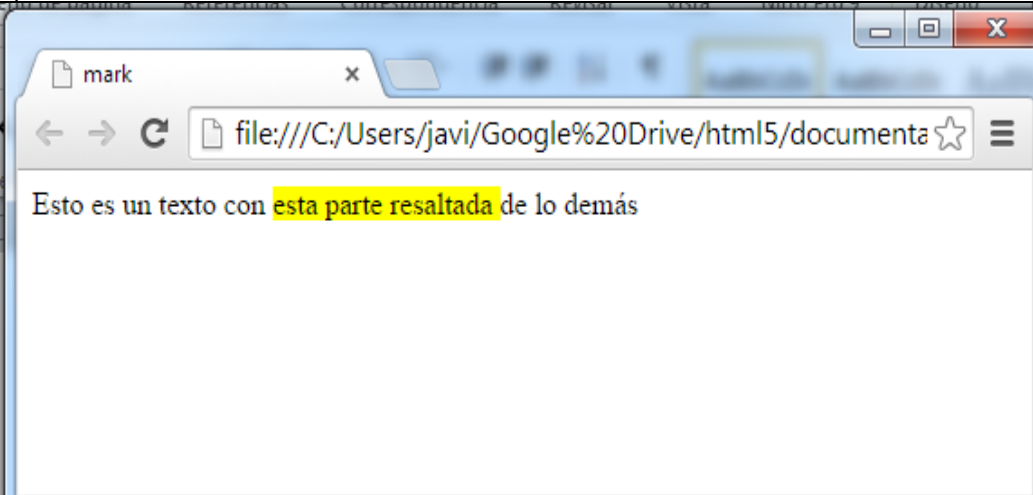
Etiqueta	<mark>				
Descripción	Resalta un texto				
Elemento	En línea				
Navegadores que la soportan y a partir de qué versión					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	6.0	4.0	9.0	11.1	5.0
Atributos	Globales y eventos				
Atributos propios	No tiene				
Diferencia entre html 4.01 y html5	No soportada por ninguna versión anterior a HTML5				

### Ejemplo:

```

<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>mark</title>
 <meta charset="UTF-8">
</head>
<body>
 Esto es un texto con <mark>esta parte resaltada </mark> de lo demás
</body>
</html>

```





*Por defecto, el color del texto es negro y se marca con el color amarillo.*

- **<time>**: Elemento que representa una fecha en formato que pueda entender el usuario y que también puede ser luego utilizada para calendarios, agendas, etc.

Etiqueta	<time>				
Descripción	Elemento que representa una fecha				
Elemento	En línea				
Navegadores que la soportan y a partir de qué versión					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	6.0	4.0	9.0	11.1	5.0
Atributos	Globales y eventos				
Atributos propios	datetime				
Diferencia entre html 4.01 y html5	No soportada por ninguna versión anterior a HTML5				

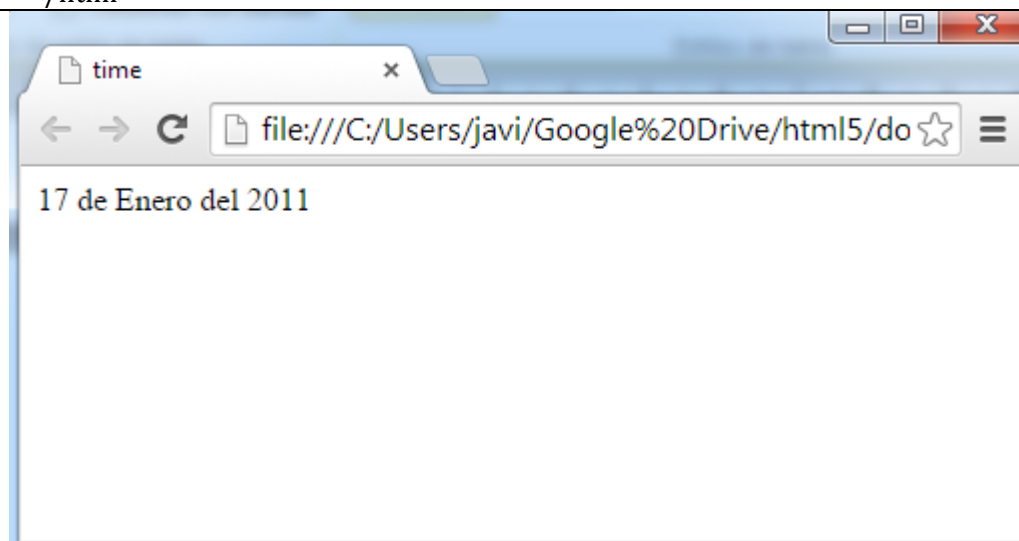
Atributos propios	Valor	Descripción
datetime     	datetime	Formato de la fecha

Ejemplo:

```

<!DOCTYPE html>
<html lang="es-es">
 <head>
 <title>time</title>
 <meta charset="UTF-8">
 </head>
 <body>
 <time datetime="2011-07-17">
 17 de Enero del 2011
 </time>
 </body>
</html>

```



**Ejercicio 27:** Vamos a trabajar con todo lo visto en este tema sólo teniendo en cuenta el contraste de colores de los elementos desde el punto de vista estético (en el siguiente

bloque cuando apliquemos estilos ya le daremos un toque mucho más vistoso a nuestros trabajos). La temática va a ser sobre MEDIOS DE TRANSPORTE y vamos a crear una estructura parecida a lo que hemos visto con los siguientes componentes:

- Un color de fondo a <body>
- Pon colores adecuados a todos los elementos de la página.
- Asigna **id** a todos elementos.
- Una cabecera <header> con:
  - o Que ocupe todo el ancho de la página y 100 píxeles de altura
  - o Un grupo de encabezados
    - Con el texto “MEDIOS DE TRANSPORTE” y el mayor de los encabezados.
    - Con el texto “Las creaciones del ser humano” con un encabezado menor que el anterior.
    - Con el texto “Página sobre trenes y aviones” con un encabezado menor que el anterior y marcando “trenes y aviones”.
- Una barra de navegación <nav> que ocupe todo el ancho de la página y 50 píxeles de altura con cuatro enlaces a dos páginas sobre trenes y otras dos sobre aviones separados por espacios en blanco (&nbsp;).
- Dos artículos <article>, uno sobre trenes y otro sobre aviones, con la siguiente estructura común:
  - o Que ocupe 600 píxeles de ancho.
  - o Que ocupe 450 píxeles de alto.
  - o Una cabecera <header> con:
    - El ancho de <article>.
    - Altura de 50 píxeles.
    - Color de fondo.
  - o Una sección <section> con:
    - El ancho de <article>.
    - Altura de 150 píxeles.
    - Una tabla con una columna y una fila sin bordes (usa el estilo `style="border: hidden;"`) con:
- El ancho de <section>.
- En la primera celda ira un elemento <figure> con un texto y una imagen sobre el tema del artículo.
- En la segunda celda un título adecuado al tema del artículo.

**Nota:** Esta manera de cuadrar elementos es obsoleta y no se emplea. Ya veremos cómo se hace realmente en el bloque de estilos.

- o Una sección <section> con:
  - El ancho de <article>.
  - Altura de 200 píxeles.
  - Un párrafo con diferentes tipos de letra y con algo del texto marcado sobre el tema del artículo.
- o Un pie <footer> con:
  - El ancho e <article>.
  - Altura de 50 pixels



- Con el texto “Fuentes de información” más de dónde has sacado la información.
- Un pie de página <footer> con:
  - Que ocupe todo el ancho de la página.
  - Altura 70 píxeles.
  - Con el nombre del autor, enlace al correo electrónico y un elemento <figure> con una imagen del autor separados por espacios en blanco y a los que se aplicará un encabezado menor que el texto normal.
- Si usas imágenes locales usa rutas relativas y la estructura que venimos usando a lo largo del curso.
- Guarda la página como ejercicio27.html

## **BLOQUE 4:**

## **CSS ES UN LENGUAJE DE ESTILO**

#### 4.1. DEFINICIÓN DE CSS3

CSS es un lenguaje de estilo o tecnología que define la presentación de los documentos HTML o XHTML. Por ejemplo, CSS abarca cuestiones relativas a tipos de letra, márgenes, color de fondo, anchura, altura y muchas más propiedades relacionadas con el aspecto de una página web.

CSS son las siglas de Cascading Style Sheets y del desarrollo de su especificación se encarga el consorcio W3C con el fin de dar unas normas que permitan simplificar la forma de presentación de la información, además de crear nuevos efectos para los distintos elementos.

	<i>La especificación oficial de CSS3 la puedes encontrar en la página <a href="http://www.w3.org/TR/CSS">http://www.w3.org/TR/CSS</a></i>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------

Este lenguaje nace para solventar la problemática en la que se vieron envueltos los diseñadores web que necesitaban cada vez más componentes para dotar a sus páginas de mejores efectos visuales que implicaban agregar más y más elementos HTML y aumentar la complejidad del lenguaje. Con CSS los aspectos estéticos se separan del contenido de los documentos HTML y no obliga a añadir más elementos al lenguaje.

	<i>HTML5 se basa totalmente en los estilos a la hora de modificar el aspecto de un elemento o formatear un texto.</i>
-------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------

	<i>Muchos elementos HTML relacionados con el aspecto de otro elemento o texto han sido eliminadas de HTML5 al igual que atributos de etiquetas que simplemente no se emplean por obsoletos.</i>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

CSS3 es la última versión de CSS y es totalmente compatible con las anteriores versiones como son CSS1 o CSS2.

La gran diferencia entre CSS3 y sus anteriores versiones es que las funcionalidades están divididas en varios documentos llamados “módulos”. Cada uno de estos módulos añade nuevas facetas a las especificadas en las anteriores versiones. Esta modularización permite mucha más flexibilidad a la hora de añadir nuevas funciones por lo que puede suceder que algún módulo pueda estar aún en fase de desarrollo.

##### 4.1.1. Ventajas de las hojas de estilo CSS

- Separación del contenido de un documento de sus aspectos estéticos, lo que permite que dicho contenido no se modifique pero si su presentación para le aplicamos algunos parámetros CSS.



- Esta separación nos permite mantener todos los aspectos estéticos dentro de una única estructura que nos servirá de base para todos nuestros trabajos además de facilitarnos las tareas de actualización.
- La reutilización de código ya que pueden definirse los mismos estilos para diferentes elementos y un archivo CSS puede valer para varias páginas web..
- CSS aporta un gran número de efectos y funcionalidades que no podría aportar por si solo cualquier lenguaje HTML.

## 4.2. INCORPORACIÓN DE ESTILOS A LOS DOCUMENTOS

Esencialmente un estilo CSS es solo una regla que le dice al navegador como desplegar un elemento concreto de una página. A la hora de agregar estilos a nuestras páginas web disponemos de tres formas básicas:

### 1. Aplicación de estilo en la etiqueta de inicio del elemento o en línea

Consiste en usar el atributo **style** de HTML, en cuyo valor se definen las reglas de estilo particulares que se aplicarán al elemento correspondiente.

Formato	Ejemplo
<code>&lt;etiqueta style="propiedad1:valor1; propiedad2:valor2; ..."&gt;</code>  <code>&lt;/etiqueta&gt;</code>	<code>&lt;p style="color:green;background-color:orange;&gt;</code>  Texto del párrafo  <code>&lt;/p&gt;</code>
En el formato vemos que el contenido de style está formado por bloques de dos datos separados por ";". Estos bloques están formados por la propiedad y su valor separados por ":"	
En el ejemplo modificamos el aspecto de un párrafo concreto poniendo el texto en color verde y el fondo en color naranja	

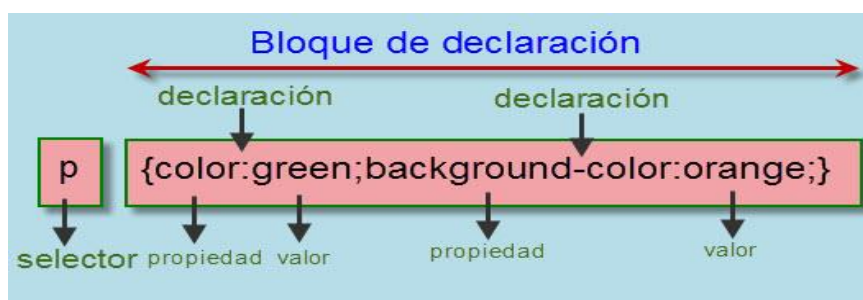
### 2. Definiendo una hoja de estilos interna

Consiste en situar la definición de las reglas de estilo que se aplican en el documento dentro de la cabecera del mismo, o sea, dentro de `<head> ... </head>` a través del elemento `<style>`. Esta lista de reglas es lo que se denomina **hoja de estilos**.

Etiqueta	<style>					
Descripción	Engloba las reglas de estilo que se aplican a un documento HTML					
Elemento	No es bloque ni en línea					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos					
Atributos	media, scoped, type					

propios	
Diferencia entre html 4.01 y html5	El atributo scoped es nuevo en HTML5

Cada regla consta de uno o más *selectores* y un *bloque de declaración* con los estilos a aplicar para los elementos del documento que se ajusten al selector. Cada bloque de declaración se define entre llaves y está formado por una o varias definiciones de estilo con el formato propiedad:valor;.



- **Selector:** El selector indica a qué elemento o elementos de una página web se aplica el estilo.

En la imagen superior se ve que el selector es p lo que indica que se van a aplicar los estilos a todos los elementos <p> del documento.

	Una misma regla de estilo puede aplicarse sobre varios selectores y un mismo selector se puede usar en varias reglas lo que implica que a un mismo elemento HTML se le pueden aplicar diferentes reglas de estilo y cada regla de estilo se puede usar en varios elementos HTML.
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- **Bloque de declaración:** La información que sigue al selector y que va entre los símbolos { (apertura) y } (cierre) incluye todas las declaraciones u opciones de formato que se van a intentar aplicar al selector.

En la imagen superior se ve que el bloque de declaración es {color:green;background-color:orange;}

- **Declaración:** Está formada por una propiedad y su valor separados por :. Cada declaración acaba en ;
  - **Propiedad:** Son las opciones de formato

- **Valor:** Es el valor de la propiedad y que tiene que ser coherente con la naturaleza de dicha propiedad, es decir, si la propiedad es un color de texto, su valor ha de ser, evidentemente, un color.

En la imagen superior se ve que el bloque de declaración consta de dos declaraciones:

- **color:green;** Que indica que el color del texto del elemento al que se va a aplicar es verde.
- **background-color:orange;** Que indica que el color de fondo del elemento al que se va a aplicar es naranja.

En resumidas cuentas, la imagen superior indica que los elementos <p> (párrafo) del documento se le aplican las reglas indicadas en el selector p y que son: color verde para el texto del párrafo y color naranja para el fondo del párrafo.

Formato	Ejemplo
<pre>&lt;head&gt;  &lt;style type="text/css"&gt;    Selector1 {propiedad1:valor1;     propiedad2:valor2; ...}    selector2 {propiedad3:valor3;     propiedad4:valor4; ...}    ..  &lt;/style&gt;  &lt;/head&gt;</pre>	<pre>&lt;head&gt;  &lt;style type="text/css"&gt;    p {      color:green;      background-color:orange;    }    h2 {      color:yellow;    }    ..  &lt;/style&gt;  &lt;/head&gt;</pre>
En el formato vemos que style, al que indicamos a través del atributo type que es una hoja de estilos, está dentro &lt;head&gt; y que está formado por bloques de dos datos separados por “;”. Estos bloques están formados por la propiedad y su valor separados por “:”   En el ejemplo modificamos el aspecto de todos los elementos &lt;p&gt;, párrafo, poniendo el texto en color verde y el fondo en color naranja y el de un encabezado &lt;h2&gt; con color amarillo para el texto	

### 3. Incorporando una hoja de estilos externa.

Es este caso guardamos las reglas de estilo siguiendo el formato que hemos visto en el punto anterior en un archivo de tipo texto, cuya extensión es normalmente .css, y lo utilizamos a través del elemento `<link>`

Etiqueta	<link>				
Descripción	Incorpora un fichero a otro según algún tipo de relación				
Elemento	No es bloque ni en línea				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos				
Atributos propios	href, hreflang, media, rel, sizes, type				
Diferencia entre html 4.01 y html5	El atributo sizes es nuevo en HTML5				

La definición de este elemento se hace dentro de `<head> .. </head>` y es de la siguiente manera:

```
<head>

 <link rel="stylesheet" type="text/css" href="fichero">

</head>
```

Los atributos y sus valores usados en `<link>` son:

- `rel="stylesheet"`: Especifica que el archivo incorporado es una hoja de estilos.
- `type="text/css"`: Indica que el contenido del archivo son reglas de estilo.
- `href="archivo"`: Indica la URL del archivo incorporado.

	<i>Pon la extensión .css a los archivos donde guardes las reglas de estilos.</i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------

Ejemplo de contenido de archivo con reglas de estilo:



```
p {
 color:green;
 background-color:orange;
}

h2 {
 color:yellow;
}
```

#### 4. Importar una hoja de estilos externa

Es similar al método anterior pero se usa el tipo especial de regla *@import* dentro de `<style> .. </style>` cuyo formato es:

```
<head>

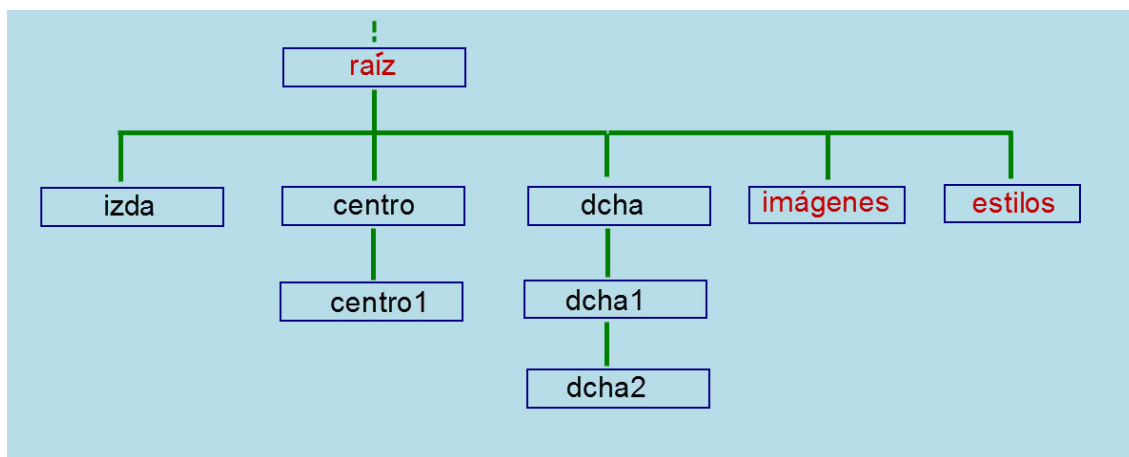
 <style type="text/css">
 @import "fichero";
 </style>

</head>
```



*Usa el método que quieras para insertar las reglas de estilo pero sería muy conveniente que utilizaras hojas de estilos externas debido a que es más fácil añadir, modificar o borrar reglas de estilo y porque se pueden usar en otros documentos HTML.*

**Ejercicio nº. 28:** Antes de hacer el siguiente ejercicio vamos a crear una nueva carpeta llamada **estilos** para guardar las hojas de estilo externas dentro de la estructura que vamos usando a lo largo del curso. Fíjate en la imagen:



Después de actualizar la estructura de carpetas vamos a hacer un sencillo ejercicio de estilos en el cual vamos a practicar los cuatro métodos de inserción de los mismos:

- Revisa los ejemplos de este apartado ya que vamos a hacer que todos los párrafos de nuestras páginas tengan un texto de color blanco y un fondo de color negro y que los encabezados <h3> tengan texto de color rojo.
- Crea la página ejercicio28.html en la carpeta “raíz” con lo siguiente:
  - El título será “Estilos”
  - Un encabezado <h1> con el texto “Ejercicio de estilos”.
  - Tres párrafos con el texto que quieras.
  - Un encabezado <h3> con el texto “Final de página”.
- Para los siguientes puntos usa de base la página creada en el apartado anterior y deja los documentos creados en la carpeta “raíz”.
- Crea la página ejercicio28-1.html aplicando los estilos del primer punto a través de la etiqueta de apertura o en línea.
- Crea la página ejercicio28-2.html aplicando los estilos del primer punto a través de una hoja de estilos interna.
- Crea la página ejercicio28-3.html aplicando los estilos del primer punto a través de una hoja estilos externa. Para ello crea el archivo estilos.css con las reglas adecuadas y déjalo en la carpeta “estilos”.
- Crea la página ejercicio28-4.html importando el archivo que has creado en el punto anterior.
- Visualiza las cuatro páginas anteriores con tus navegadores y deberás ver el mismo resultado en las cuatro.

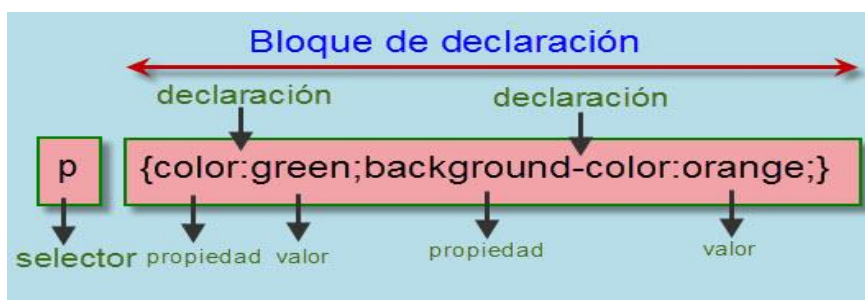
### 4.3. SELECTORES

Tal y como hemos visto, los selectores permiten identificar los elementos del documento HTML al que se va aplicar el estilo definido dentro del bloque de declaración. Este identificador sigue las siguientes reglas sintácticas:

- Empieza por un carácter alfabético (A..Z ó a..z).
- Después de ese primer carácter puede haber más caracteres alfabéticos, guiones ( - ) o números (0-9).

Este identificador puede ir precedido por un carácter “#” que sirve para los selectores de atributo **id** o por un “.” para los selectores de clase o por un nombre de elemento para los selectores de elementos.

	<i>Los selectores no se usan en la definición de estilos en línea, es decir, en los que se definen en la etiqueta de apertura, en todos los demás sí.</i>
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------



## 4.4. HERENCIA Y CASCADA

Cuando se carga un documento HTML en un navegador, éste construye un árbol de contenido (DOM) que relaciona todos sus elementos e indica cómo se comportan. Dichos elementos tienen, normalmente, una relación de parentesco entre sí (padre, hijo, hermano, hermano adyacente o descendiente) que marca su comportamiento ya que puede heredar alguna propiedad de los elementos superiores o transmitir otras propiedades a sus elementos descendientes.

También puede suceder que los valores de algunas propiedades entren en conflicto dependiendo de su aplicación por lo que habrá que tener un criterio claro de cual se aplica y en qué caso (cascada).

### 4.4.1. Herencia

La herencia en CSS es la forma mediante la cual algunas propiedades de un elemento de nivel superior se transmiten a los elementos de nivel inferior que pertenecen a su contexto.

Todos los elementos de una página HTML heredan alguna de las propiedades de su elemento inmediatamente superior a excepción del elemento <html> que no tiene elemento padre.

Un ejemplo de propiedad que se hereda es el color de fondo de un elemento (background-color).

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Herencia</title>

 <meta charset="utf-8">

<style>

 .fondoverde

 {background-color:#ccffcc;}

 #articulo2

 {background-color: grey;}

</style>

</head>
```

```

<body class="fondoverde">

<article>

 <h1> Artículo 1</h1>

</article>

<article id="articulo2">

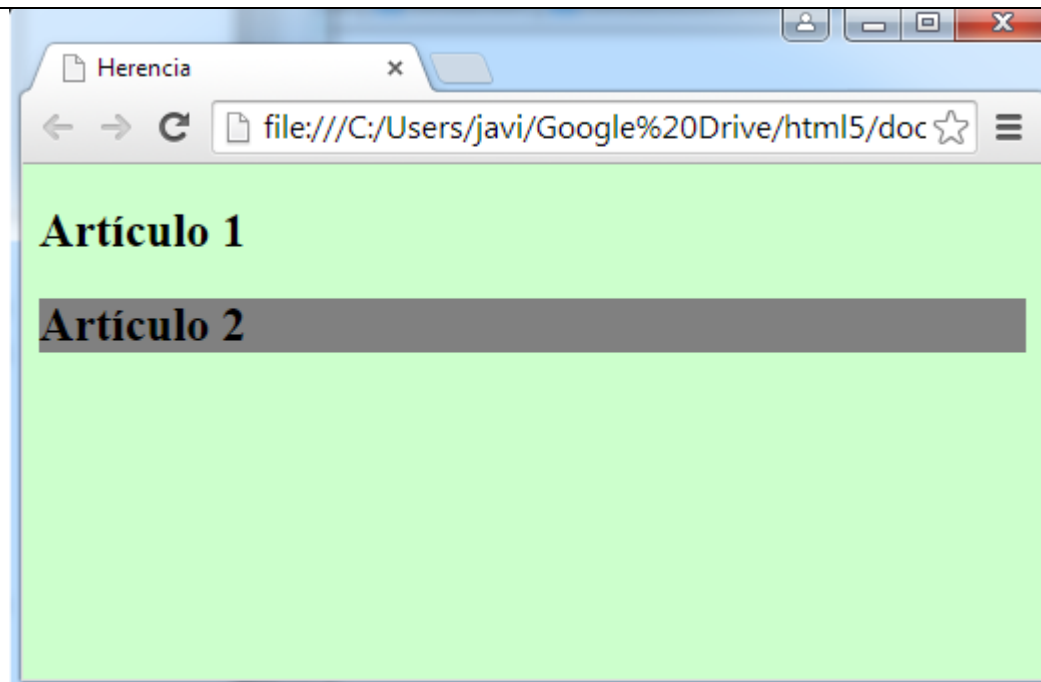
 <h1> Artículo 2</h1>

</article>

</body>

</html>

```



En este ejemplo se ve que el primer artículo no tiene asignado color de fondo (background-color) por lo que lo hereda de <body>. En cambio, el segundo artículo tiene otro color de fondo debido a que le ha sido asignado.

Un ejemplo de propiedad que no se hereda es el fondo del elemento (background) pero se puede forzar a que un elemento herede de su antecesor una propiedad el valor *inherit* asignado al selector de estilo.

```

<!DOCTYPE html>

<html lang="es-es">

```

```
<head>

 <title>Sin Herencia</title>

 <meta charset="utf-8">

<style>

 .fondoverde

 {background: url("html5jpg.jpg") no-repeat center;}

 .art

 {background: inherit;

 width: 200px;

 height: 200px;

 }

 .articulo

 {background-color: #89FF6b;}

</style>

</head>

<body class="fondoverde">

<article class="articulo">

 <h1> Artículo 1</h1>

</article>

<article class="articulo">

 <h1> Artículo 2</h1>

</article>

<article class="articulo">

 <h1> Artículo 3</h1>
```

```

</article>

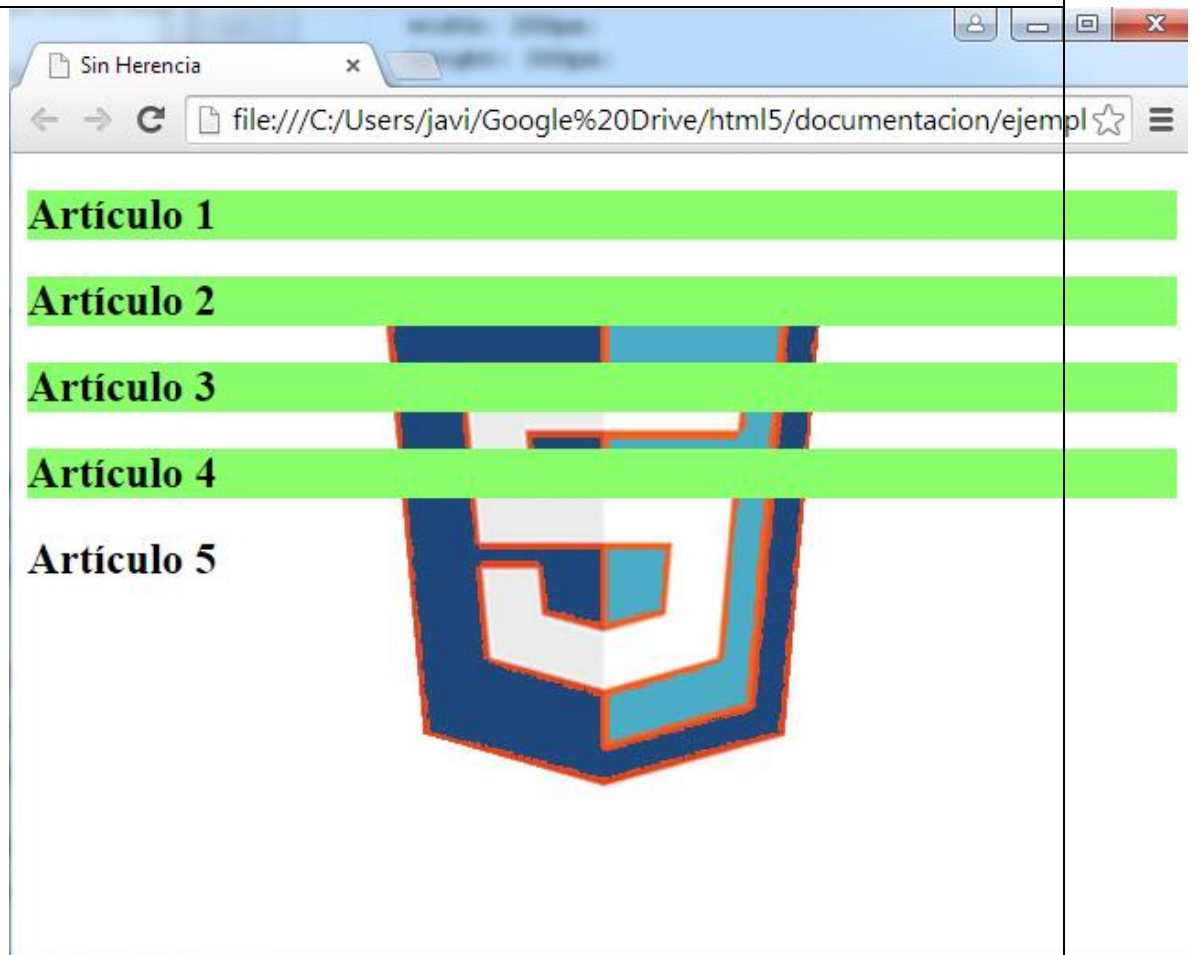
<article class="articulo">
 <h1> Artículo 4</h1>
</article>

<article class="art">
 <h1> Artículo 5</h1>
</article>

</body>

</html>

```



En este ejemplo se ve que se definió una imagen de fondo al documento (elemento `<body>`) y se ve que no se hereda al resto de elementos hijos de `<body>` como son los elementos `<article>`

```

<!DOCTYPE html>

<html lang="es-es">

```

```
<head>

 <title>Sin Herencia</title>

 <meta charset="utf-8">

<style>

 .fondoverde

 {background: url("html5jpg.jpg") no-repeat center;}

 .art

 { background:inherit;

 width: 200px;

 height: 200px;

 }

 .articulo

 {background-color: #89FF6b;}

</style>

</head>

<body class="fondoverde">

<article class="articulo">

 <h1> Artículo 1</h1>

</article>

<article class="articulo">

 <h1> Artículo 2</h1>

</article>

<article class="articulo">

 <h1> Artículo 3</h1>

</article>

<article class="articulo">

 <h1> Artículo 4</h1>
```



```

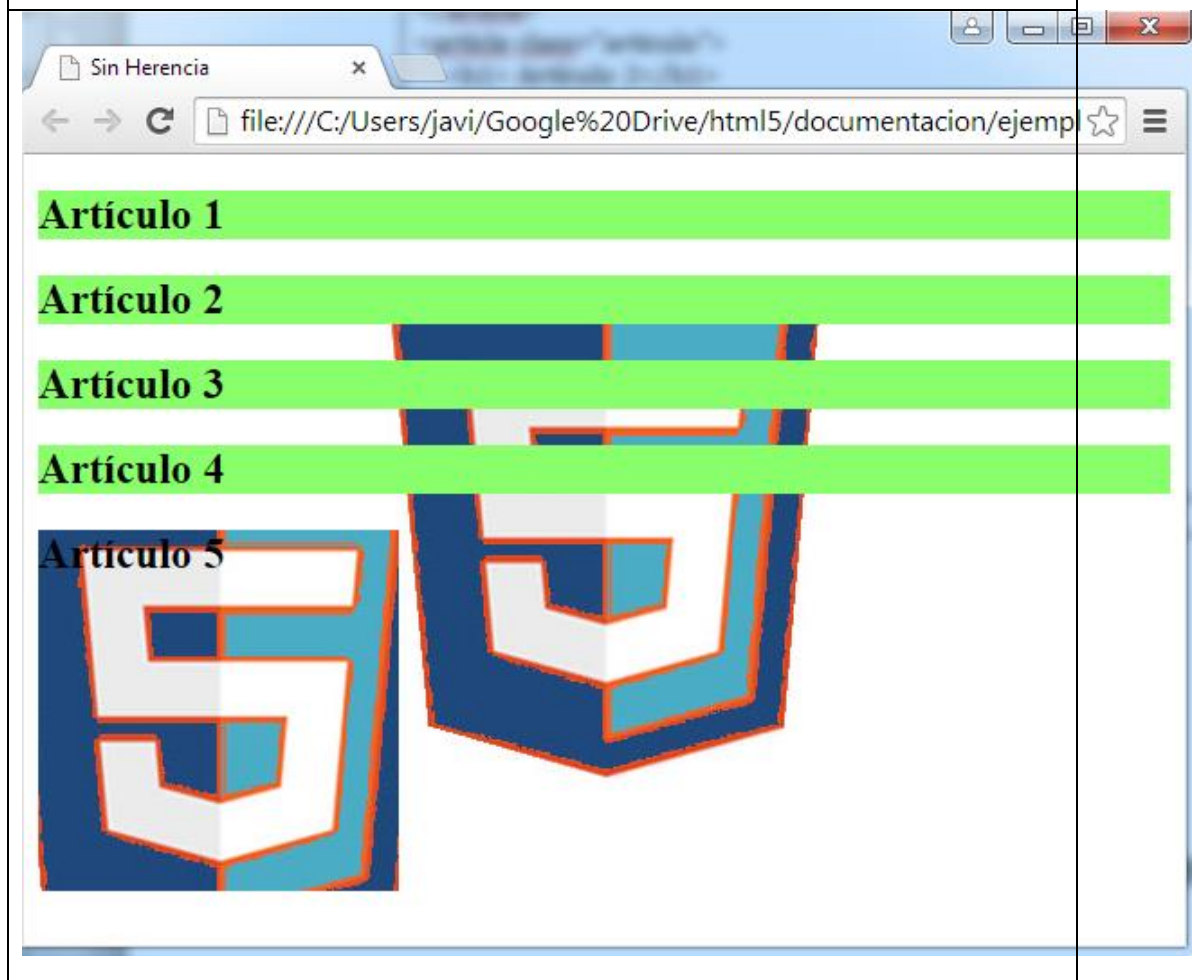
</article>

<article class="art">
 <h1> Artículo 5</h1>
</article>

</body>

</html>

```



En cambio, en este ejemplo, la clase `.art` indica que hereda el fondo del elemento padre (`background:inherit;`) lo que hace que se le asigne al elemento `<article>` con el texto “Artículo 5” el fondo de su padre, o sea, de `<body>`.

#### 4.4.2 Cascada

La cascada en CSS resuelve el problema de qué hacer en el caso aplicar las declaraciones de CSS contradictorias sobre algún elemento. Se basa en la importancia (donde se ha declarado), especificidad (selectores más concretos a más globales) y orden (la última declaración prevalece sobre la primera), es decir, a declaraciones CSS iguales se decide por el grado de especificidad y si ese grado es el mismo por el orden de declaración.

## Preferencia por especificidad de estilos en cascada



la definición in-line tiene la preferencia más alta

### Ejercicios:

Toma como base el archivo avanzado.html, creado en la sección anterior, y elimina la referencia al archivo de estilos creado en los ejercicios de dicha sección guardándolo después como “herencia.html”.

Fíjate bien en el diagrama que has hecho en el ejercicio 36 para que veas como influye la herencia en los elementos HTML5 y vas a crear el archivo “herencia.css” en la carpeta “estilos” para los estilos externos que vas a utilizar.

### Ejercicio 38:

- Asigna el id “sección1” al primer elemento <section> y el id “marca1” al primer elemento <mark> del primer elemento <p> del elemento identificado con “sección1”

En cada uno de los puntos siguientes indica a que elementos afecta los estilos que hayas asignado para que ves cuales se heredan, cuales no y que elementos permiten dicha herencia.

- Agrega todos los estilos al documento “herencia.css”
- Incorpora la hoja de estilos anterior a la página herencia.html
- Crea la clase “colordefondo” que pondrá el color de fondo #FFCC66 a los elementos que se aplique. Asigna dicha clase a <body>.
- Crea la clase “colordetexto” que pondrá el texto en color “orange” y aplícalo a “seccion1”.

### Ejercicio 39:

- Para practicar con el concepto de “cascada”, dejamos el archivo “herencia.css” como está y vamos a agregar todos los estilos al elemento <style type=“text/css”> ... </style> que vas a incorporar al archivo herencia.html.
- Crea el estilo “colordefondolocal” que pondrá el color de fondo #66FFCC al elemento <body>. ¿Qué estilo prevale? ¿El de la clase o el asignado al elemento?.
- Crea la clase “clasecolor” que pondrá el color de fondo #CC66FF a los elementos que se aplique y agrégala a <body>, es decir, ahora <body> tiene dos clases que asignan el mismo atributo con diferente valor. ¿Cuál prevale?.

- Cambia el orden en que hayas puesto las clases en `<body>` y mira a ver si influye.
- Agrega a la etiqueta `<body>` el parámetro “style” y, a través de este parámetro, indica que el color de fondo es verde. ¿Cuál de los estilos prevalece?.
- Elimina estilo que has agregado en el punto anterior, es decir, en el que indicas que `<body>` tiene color de fondo verde.

**Ejercicio 40:**

- Agrega a “herencia.css” un estilo que indique que todos los elementos `<mark>` tengan color de fondo negro y color de texto blanco y la clase “clasecolor” que indica que el color de texto es #333333.
- Agrega un estilo que indique que todos los elementos `<mark>` tengo color de texto amarillo dentro del elemento `<style>` del documento. ¿Cuál prevalece?.
- Aplica al elemento “marca1” la clase “clasecolor”. ¿Cuál prevalece?
- Agrega un estilo que asigne al elemento con `id=“marca1”` el color rojo. ¿Cuál prevalece?.
- Agrega un estilo a la etiqueta de apertura de “marca1” y pon en ese estilo que el color de texto es magenta. ¿Cuál prevalece?.

## 4.5. UNIDADES DE MEDIDA

En toda página web necesitamos manejar con soltura y buen criterio las dimensiones (anchura, altura, márgenes y tamaños) de los elementos HTML ya que el aspecto estético de un documento HTML, sobre todo, va a depender muy mucho de cómo se visualicen dichos elementos en conjunto. Para ello necesitamos conocer los tipos y unidades de medida que podemos emplear en nuestras páginas a través de CSS3.

A todas las medidas se les asigna un valor numérico entero o decimal positivo, aunque puede ser negativo en algunos casos, seguido de una unidad de medida sin espacios en blanco entre medio.

**Ejemplos:** margin: 1px; border: 0.5in; width:25%;

Las medidas en CSS3 se basan en diferentes unidades de medida (píxeles, centímetros, etc.) y en tres tipos de uso (unidades absolutas, unidades relativas y de porcentaje).

Unidad de medida					
em, ex, %, px, cm, mm, in, pt, pc	1.0	3.0	1.0	1.0	3.5
ch	27.0	9.0	1.0	7.0	20.0
rem	4.0	9.0	3.6	4.1	11.6
vh, vw	20.0	9.0	19.0	6.0	20.0
vmin	20.0	9.0*	19.0	6.0	20.0
vmax	26.0	No soportado	19.0	No soportado	20.0

(\*) Internet Explorer 9 usa vmin con el nombre vm que no es estándar para el resto de navegadores

### 4.5.1. Unidades Absolutas

Una unidad absoluta es aquella que indica que la medida es invariable, es decir, que no depende de ni ningún otro valor ni varía.

La principal ventaja de este tipo de unidades es que no se necesita ningún tipo de cálculo para utilizarla pero tiene la desventaja de que es un sistema muy poco flexible y difícilmente adaptable de diseño o de visualización en diferentes medios.

Unidad	Descripción
cm	centímetros
mm	milímetros

in	pulgadas (1in = 2.54vcm)
pt	puntos (1pt = 0,0353 cm)
pc	picas (1pc = 12 pt)

#### 4.5.2. Unidades relativas

Una unidad relativa es aquella que puede variar ya que su valor va a depender de otros valores que normalmente van a venir determinados por los elementos de niveles superiores que contengan dentro de su contexto al elemento que vamos a aplicar dichas unidades.

Estas unidades son las más utilizadas por su flexibilidad tanto a la hora de diseño como de visualización en diferentes medios ya que mantienen las proporciones entre los diferentes elementos.

Unidad	Descripción
em	Relativa al tamaño del texto del element (2em significa 2 veces el tamaño del texto actual)
ex	Relativa a la altura del eje x de la fuente actual (apenas se usa)
ch	Relativa a la anchura del carácter “0” (cero) de la fuente usada
rem	Relativa al tamaño del texto del elemento raíz
vw	Relativa al 1% del ancho del dispositivo donde se visualiza la página
vh	Relativa al 1% del alto del dispositivo donde se visualiza la página
vmin, vm	Relativa al menor tamaño, 1% de ancho o 1% de alto, del dispositivo donde se visualiza la página
vmax	Relativa al mayor tamaño, 1% de ancho o 1% de alto, del dispositivo donde se visualiza la página
%	Relativa un tanto por ciento al tamaño del elemento contenedor
px	pixels

#### 4.5.3. Porcentajes

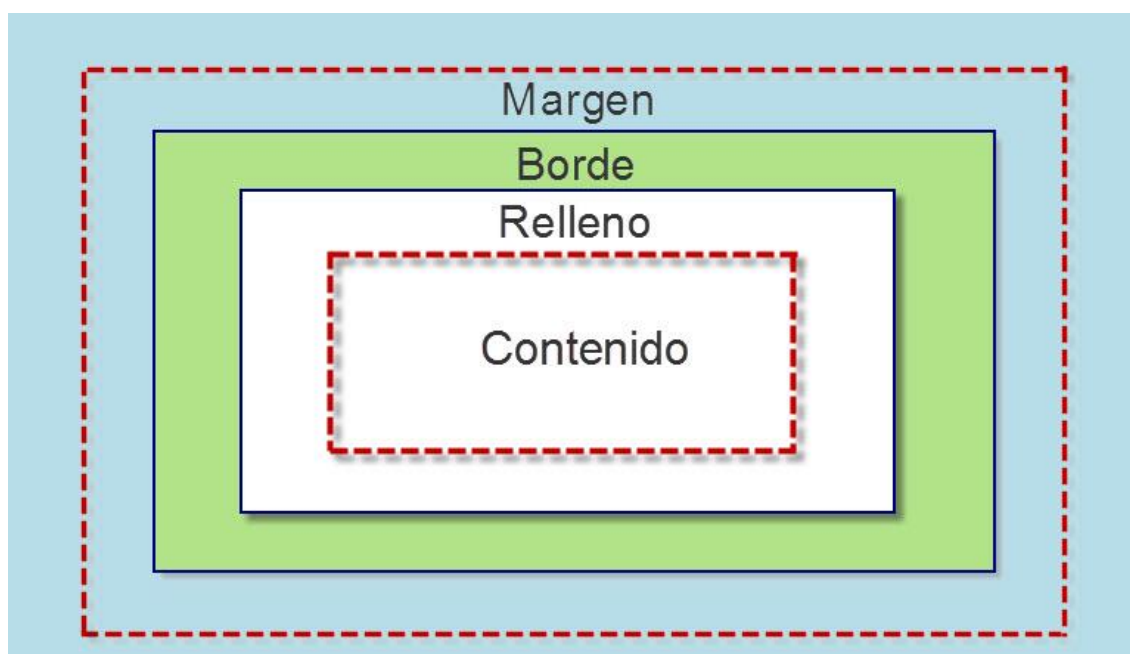
Es una unidad de medida relativa representada por un valor numérico seguida del símbolo % y que es, normalmente, un tanto por ciento de la unidad de medida correspondiente del elemento contenedor.

## 4.6. MODELO DE CAJAS

El modelo de cajas o “box model” se basa en representar, a través de CSS, **todos los elementos** de los documentos HTML como cajas rectangulares que se crean automáticamente al definir cada elemento.

Las partes que tiene un caja son:

- **Contenido** (content): Es el contenido del elemento HTML (texto de un elemento <h1>, texto de un elemento <p>, etc.)
- **Relleno** (padding): Espacio que hay entre el borde y el contenido.
- **Borde** (border): Línea que circunda todo el elemento incluyendo el contenido y su relleno.
- **Margen** (margin): Espacio en blanco que se define alrededor del borde.
- **Fondo** (background): El color o imagen de fondo que tiene elemento.



### 4.6.1 Dimensiones

En este punto vamos a ver las propiedades CSS3 que marcan tanto el ancho de un elemento (width) como su altura (height).

Propiedad	Descripción	Posibles valores
height	Define la altura de un elemento	auto, <i>altura</i> , %, inherit
max-height	Define la altura máxima de un elemento	none, <i>altura</i> , %, inherit
max-width	Define el ancho máximo de un elemento	none, <i>anchura</i> , %, inherit
min-height	Define la altura mínima de un elemento	<i>Altura</i> , %, inherit

min-width	Define el ancho mínimo de un elemento	<i>Anchura, %, inherit</i>
width	Define el ancho de un elemento	auto, <i>anchura, %, inherit</i>

Valor	Descripción
auto	El navegador calcula el valor
<i>Anchura, Altura</i>	Especifica un valor numérico seguido de una unidad de medida como px, in, cm, etc. El valor por defecto es 0px
%	Especifica un tanto por ciento con relación al valor de la misma propiedad del elemento contenedor
inherit	Especifica que el valor se heredará del elemento padre
none	Ningún valor

**Ejemplo:**

```

<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>Altura y anchura</title>
 <meta charset="UTF-8">
 <style type="text/css">
 .fondoverde
 {background-color:#ccffcc;}

 section
 {background-color:orange;
 width: 510px;
 height: 1in;
 }
 </style>

```

```
</head>

<body class="fondoverde">

 <h1>Dimensiones</h1>

 <section>

 Texto de la sección

 Texto de la sección

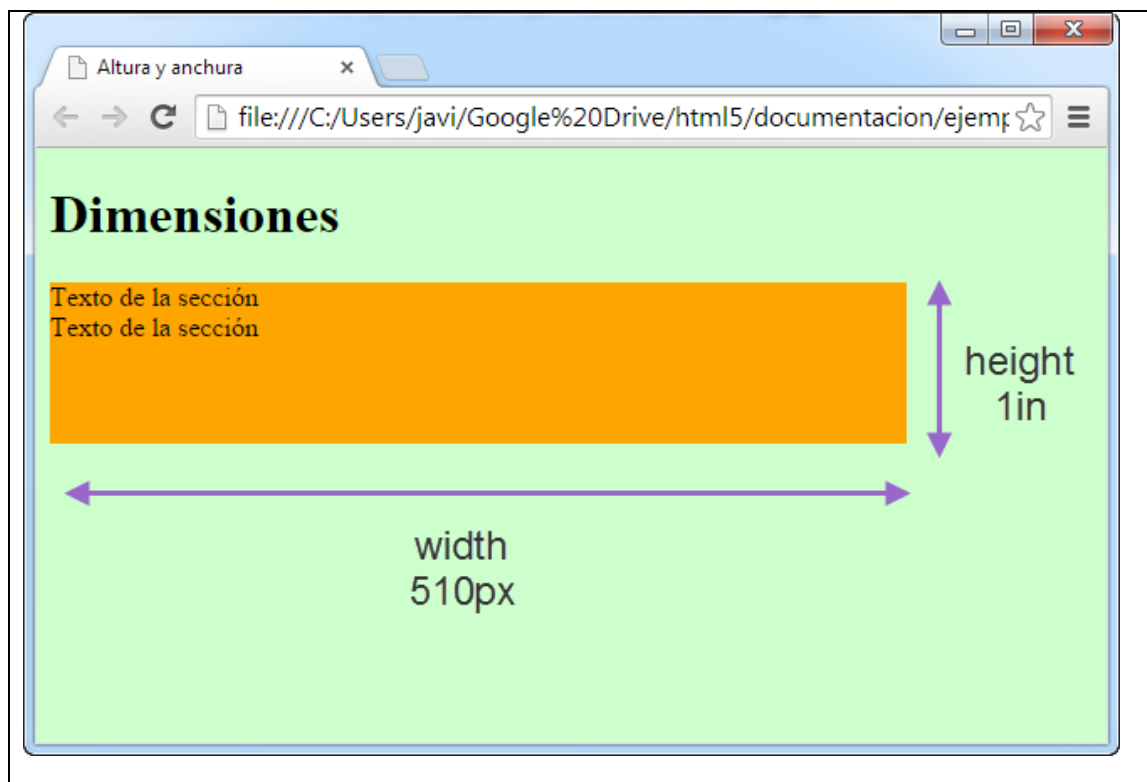
 </section>

</body>

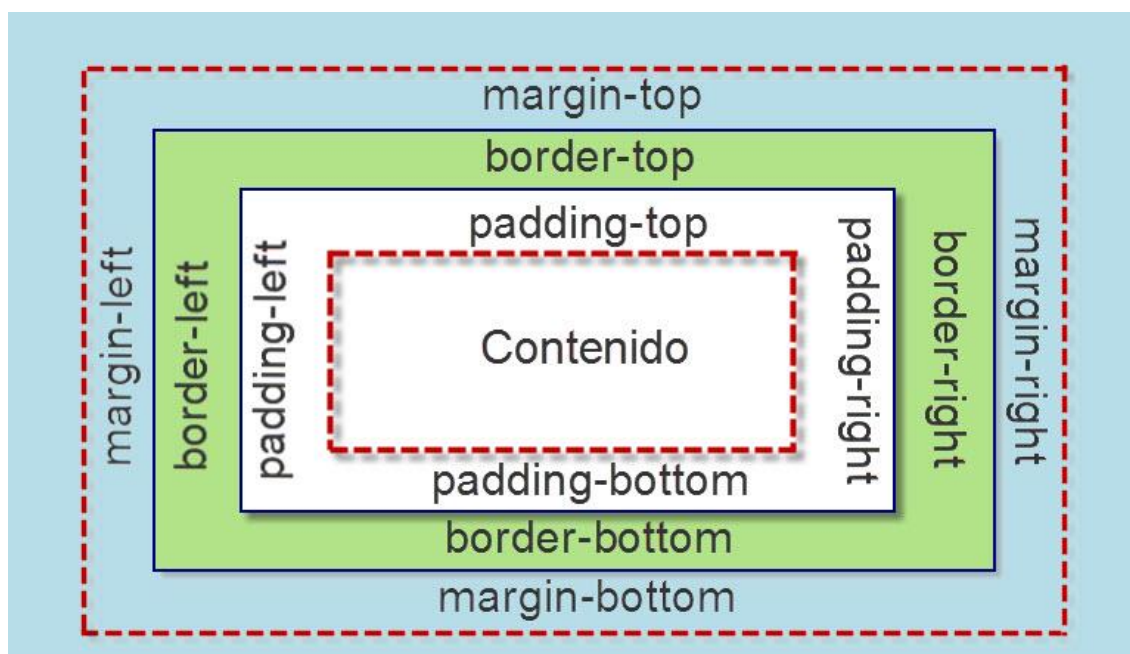
</html>
```







En el ejemplo asignamos 1 pulgada de altura y 510 pixels de ancho al elemento `<section>`.



#### 4.6.2. Propiedad “Sintaxis alternativa” (Shorthand)

Esta propiedad permite asignar de una manera simple y única los valores más importantes de las propiedades de un elemento. Esta asignación se hace a través de un selector seguido de los valores de las propiedades separados por espacios y en el orden: *superior* (top), *derecho* (right), *bottom* (inferior) e *izquierdo* (left).

Como ejemplo fíjate en el siguiente apartado donde se aplica la propiedad “Sintaxis alternativa” al margen de una caja.

#### 4.6.3. Margen

Espacio que hay alrededor de un elemento por encima de borde. Este espacio no tiene color de fondo y es completamente transparente.

Propiedad (*)	Descripción
margin	Propiedad sintaxis alternativa
margin-bottom	Define el espacio del margen inferior
margin-left	Define el espacio del margen izquierdo
margin-right	Define el espacio del margen derecho
margin-top	Define el espacio del margen superior

(\*) Los valores de las propiedades anteriores son iguales a los vistos en el apartado de

#### 4.6.1. dimensiones.

**Ejemplo:**

```
section
{
 margin-top: 20px;
 margin-right: 35px;
 margin-bottom: 40px;
 margin-left: 60px;
}
```

En este ejemplo se indica que el elemento <section> tiene un margen superior de 20 pixels, un margen inferior de 40 pixels, un margen derecho de 35 pixels y un margen izquierdo de 60 pixels (fíjate en la imagen superior para que ubiques correctamente cada una de las propiedades).

#### - Propiedad “Sintaxis alternativa” del margen de una caja

Esta propiedad se asigna a través de la propiedad “margin”.

**Ejemplos:**

```
section
{
 margin: 20px 35px 40px 60px;
}
```

Este ejemplo sería el equivalente al ejemplo superior.

```
section
{
 margin: 20px;
}
```

En este ejemplo todos los márgenes (izquierdo, derecho, superior e inferior) tienen 20 pixels.

```
section
{
 margin: 20px 40px;
}
```

En este ejemplo los márgenes superior e inferior valen 20 pixels y los márgenes izquierdo y derecho valen 40 pixels.

```
section
{
 margin: 20px 40px 35px;
}
```

En este ejemplo el margen superior vale 20 pixels, los márgenes izquierdo y derecho valen 40 pixels y el margen inferior 35 pixels.

#### 4.6.3. Borde

El borde de una caja o línea que rodea tanto el relleno como el contenido de un elemento se puede indicar su anchura, color y estilo a través de propiedades CSS3 tanto de forma individual (una caja tiene cuatro bordes) como global.

- **Estilo:** Con la propiedad *border-style* especificamos el tipo de borde del elemento (línea de puntos, línea continua, etc.)

	Ninguna de las propiedades del borde de un elemento tendrá efecto sino se define la propiedad <b>border-style</b>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------

Algunos Valores de <i>border-style</i>	Descripción
none	Sin borde
dotted	Define un borde de puntos
dashed	Define un borde de guiones
solid	Define un borde de línea continua

*border-style* también tiene la propiedad “sintaxis alternativa”.

- **Color:** Con la propiedad *border-color* se define el color del borde
- **Anchura:** Con la propiedad *border-width* determinamos el ancho del borde. Este puede expresarse en pixels o con los valores predefinidos; thin (fino), medium (medio) o thick (grueso). También dispone de la propiedad de “sintaxis alternativa”.

Propiedad	Descripción
<u>border</u>	Sintaxis alternativa del borde general
<u>border-bottom</u>	Sintaxis alternativa del borde inferior
<u>border-bottom-color</u>	Define el color del borde inferior
<u>border-bottom-style</u>	Define el estilo del borde inferior
<u>border-bottom-width</u>	Define el ancho del borde inferior
<u>border-color</u>	Define el color de los cuatro bordes
<u>border-left</u>	Sintaxis alternativa del borde izquierdo

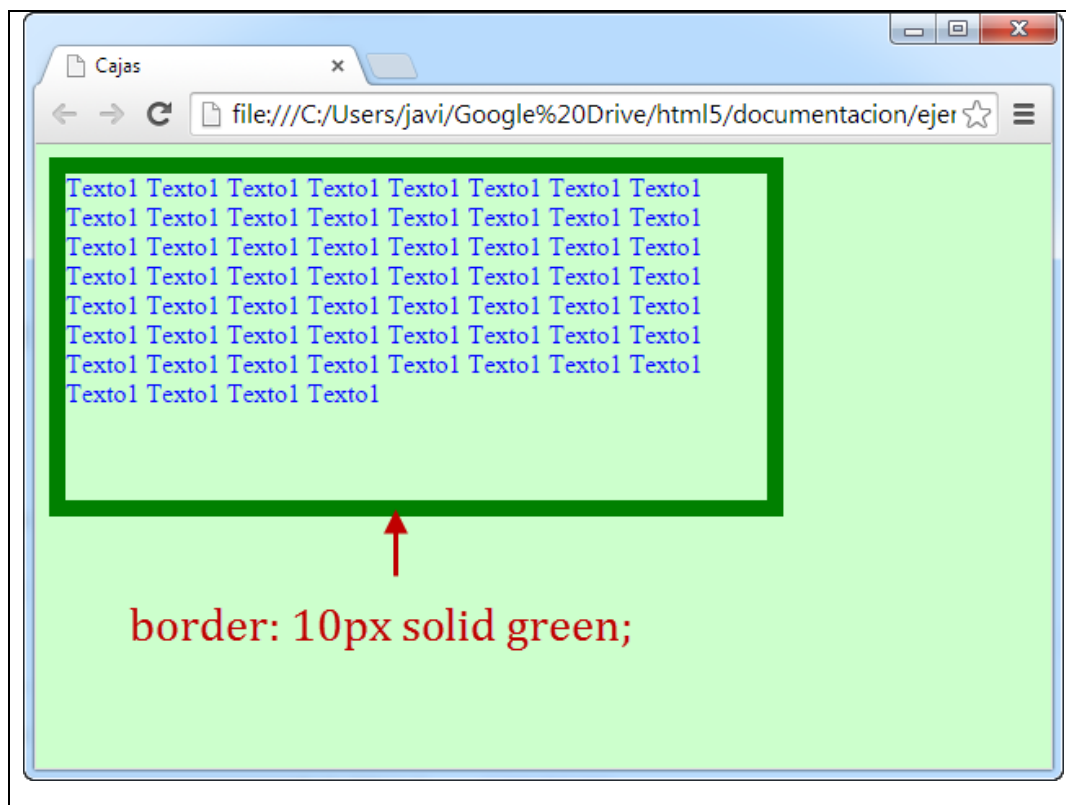
<u><a href="#">border-left-color</a></u>	Define el color del borde izquierdo
<u><a href="#">border-left-style</a></u>	Define el estilo del borde izquierdo
<u><a href="#">border-left-width</a></u>	Define el ancho del bore izquierdo
<u><a href="#">border-right</a></u>	Sintaxis alternative del borde derecho
<u><a href="#">border-right-color</a></u>	Define el color del borde derecho
<u><a href="#">border-right-style</a></u>	Define el estilo del borde derecho
<u><a href="#">border-right-width</a></u>	Define el ancho del borde derecho
<u><a href="#">border-style</a></u>	Sintaxis alternativa del estilo de los bordes
<u><a href="#">border-top</a></u>	Sintaxis alternativa del borde superior
<u><a href="#">border-top-color</a></u>	Define el color del borde superior
<u><a href="#">border-top-style</a></u>	Define el estilo del borde superior
<u><a href="#">border-top-width</a></u>	Define el ancho del borde superior
<u><a href="#">border-width</a></u>	Define el ancho de los cuatro bordes

#### - Propiedad de “sintaxis alternativa” del borde de un caja

La propiedad “sintaxis alternativa” del borde de una caja define los tres parámetros anteriores y en este orden: border-width, border-style (requerido) y border-color.

##### Ejemplo:

```
section {
 border: 10px solid green;
}
```



En este ejemplo se define un borde de 10 pixels de grosor por los cuatro lados, con línea continua y de color verde.

#### 4.6.4. Relleno

Es el espacio que hay entre el borde y el contenido de un elemento. No se le asigna color propio pero si le afecta el color de fondo (background-color) del elemento.

Propiedad	Descripción
padding	Propiedad sintaxis alternative
padding-bottom	Define el espacio del relleno inferior
padding-left	Define el espacio del relleno izquierdo
padding-right	Define el espacio del relleno derecho
padding-top	Define el espacio del relleno superior

Valor	Descripción
<i>Longitud</i>	Especifica un valor numérico seguido de una unidad de medida como px, in,cm, etc. El valor por defecto es 0px
%	Especifica un tanto por ciento con relación al valor de la misma propiedad del elemento contenedor

**Ejemplo:**

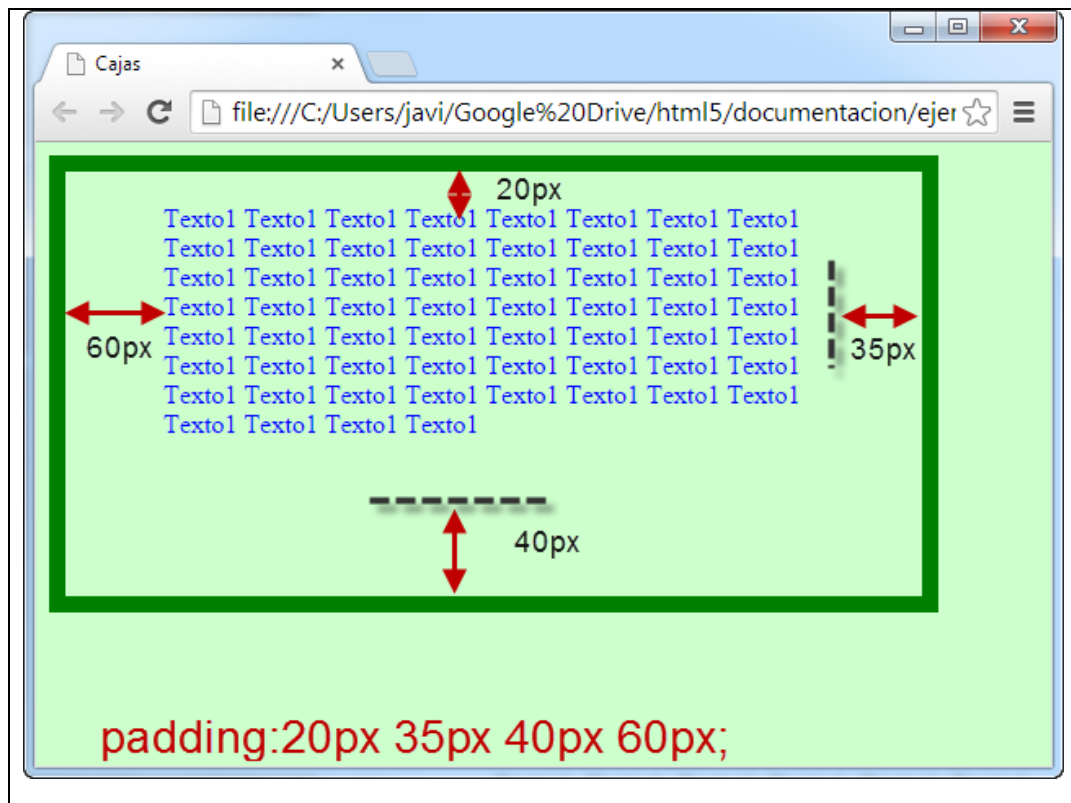
```
section
{
padding-top: 20px;
padding-bottom: 40px;
padding-right: 35px;
padding-left: 60px;
}
```

**- Propiedad “sintaxis alternativa” del relleno de una caja**

Esta propiedad se asigna a través de la propiedad “padding”.

**Ejemplos:**

```
section
{
padding: 20px 35px 40px 60px;
}
```



En este ejemplo se indica que el elemento `<section>` tiene un relleno superior de 20 pixels, un relleno inferior de 40 pixels, un relleno derecho de 35 pixels y un relleno izquierdo de 60 pixels (fíjate en la imagen superior para que ubiques correctamente cada una de las propiedades)

#### 4.6.5 Fondos

Es una propiedad estética que sirve para poner un color de fondo o una imagen en la caja del elemento.

El fondo solo se visualiza en el área de contenido y el relleno.

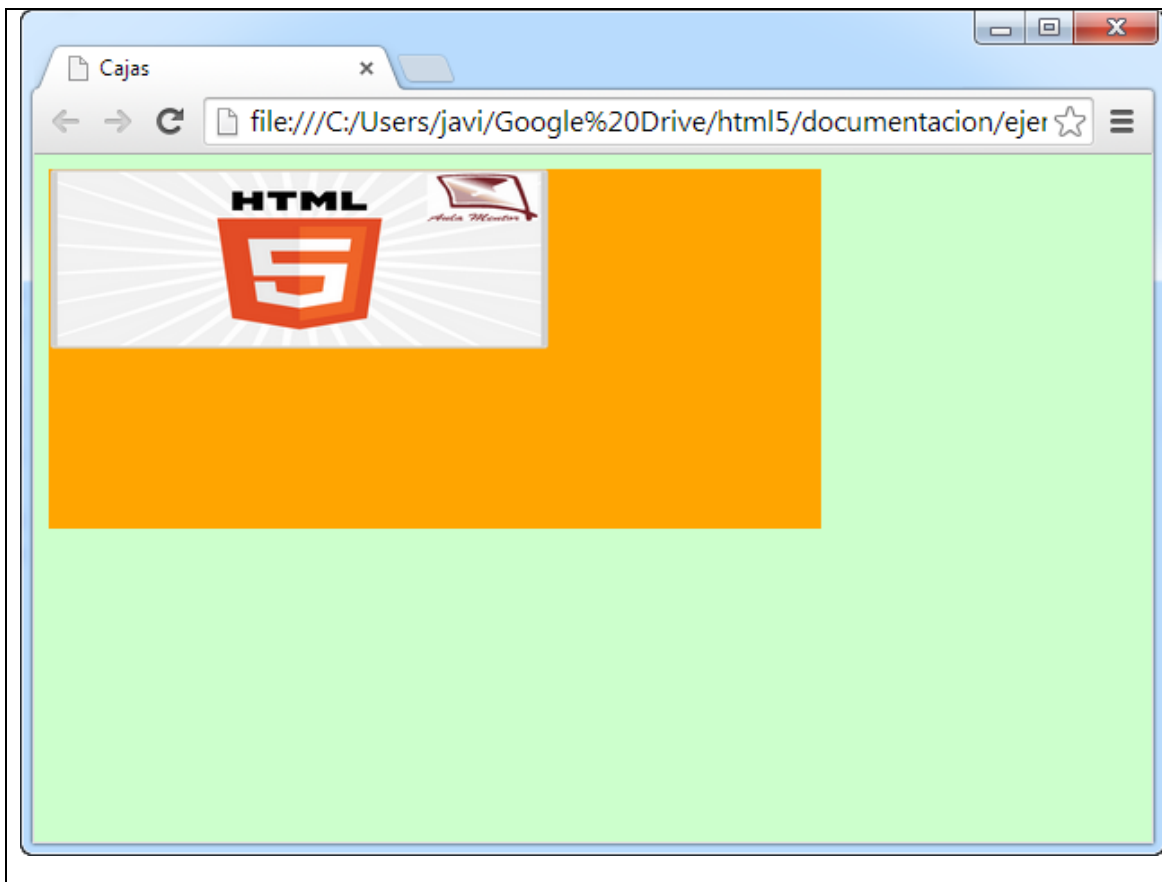
Se pueden indicar un color de fondo y una imagen a la vez para mostrar el color en el caso de que haya algún problema para visualizar la imagen.

Si la imagen que se pone de fondo es demasiado grande solo se ve la parte que permitan las dimensiones del elemento. En el caso de usar imágenes más pequeñas que la dimensión del elemento se puede repetir tanto horizontal como verticalmente hasta cubrir toda esa dimensión.



Propiedad	Descripción	Valores
<u>background</u>	Sintaxis alternativa	background-color background-position background-size background-repeat background-origin background-clip background-attachment background-image initial inherit
<u>background-attachment</u>	Define si una imagen de fondo está fija o se mueve cuando se hace scroll con el resto de la página	Scroll, fixed, local, initial, inherit
<u>background-color</u>	Define el color de fondo de un elemento	<i>Color</i> , transparent, initial, inherit
<u>background-image</u>	Indica la imagen de fondo de un elemento	url('URL'), none, initial, inherit
<u>background-position</u>	Indica la posición de la imagen de fondo	left top, left center, left bottom, right top, right center, right bottom, center top, center center, center, bottom, x%, y%, <i>xpos ypos</i> , initial, inherit
<u>background-repeat</u>	Indica si se repite o no una imagen de fondo	Repeat, repeat-x, repeat-y, no-repeat, initial, inherit

### Ejemplo:



En este ejemplo definimos una sección de ancho de 430 pixels, una altura de 200 pixels (width, height), fondo de color naranja (background:orange), color de texto azul (color:blue), con la imagen “mentorhtml5.png” (background:url(“mentohtml5.png”) sin repeticiones (background:no-repeat) y posicionada en la esquina superior izquierda (background: left top) y que ocupa el 65% del ancho y el 50% de la altura (background-size: 65% 50%) del elemento <section>.

### Ejemplo completo:

Vamos a hacer una caja <section> con todos los parámetros que hemos empleado anteriormente y ver cómo queda.

```
<!DOCTYPE html>

<html>

 <head>

 <title>Cajas</title>

 <meta charset="UTF-8">

 <style type="text/css">
```

```
.fondoverde
{background-color:#ccffcc; }

section
{
 border: 10px solid green;
 padding: 20px 35px 40px 60px;
 width:430px;
 height:200px;
 background: orange url("mentorhtml5.png") no-repeat left top;
 background-size: 65% 50%;
 color: blue;
}

</style>
</head>
<body class="fondoverde">

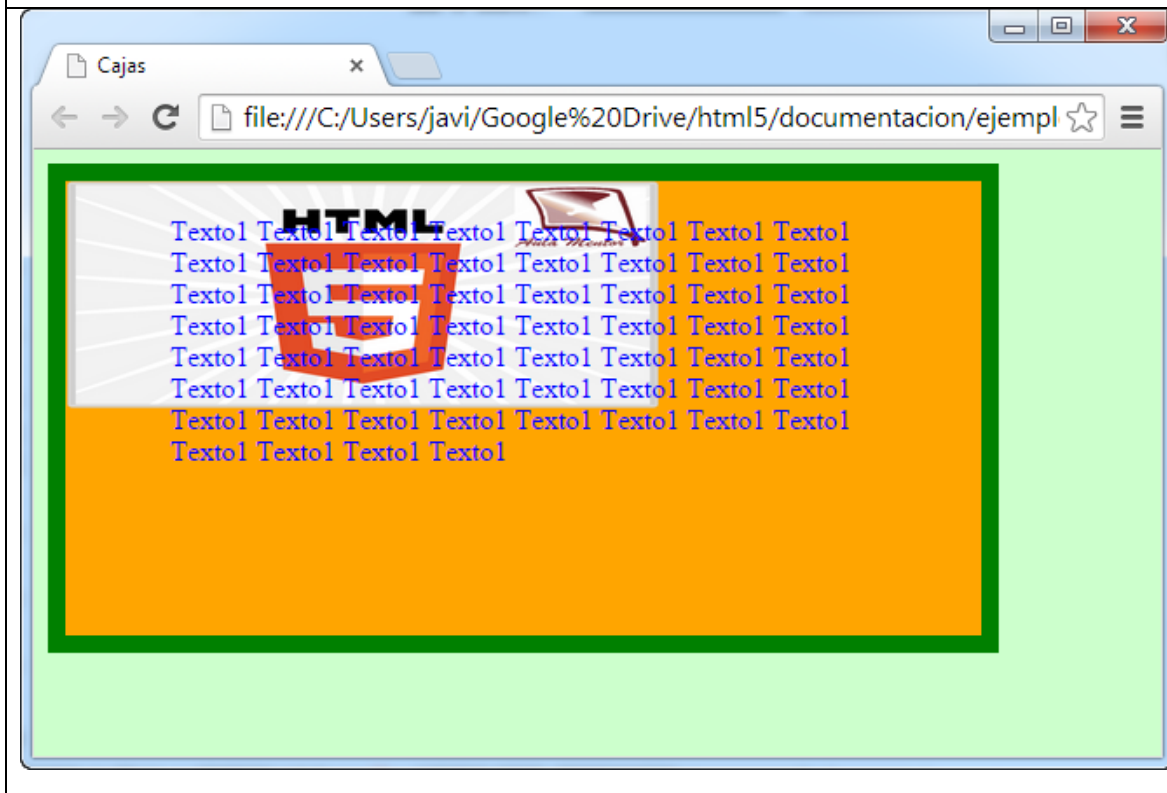
<section>

 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1

</section>
```

```
</body>
```

```
</html>
```



### Ejercicios:

Tomando como base el archivo del ejercicio 36, vamos a crear cajas para algunos de los elementos que forman el documento html.

Guarda el archivo anterior como cajas.html dentro de la carpeta tema4 de raíz y crea el archivo de estilos cajas.css en la carpeta estilos que es donde vas a guardar los estilos de estas cajas ya que no debes poner ninguno en el documento html.

Pon los **id** a los elementos, crea clases y utiliza los selectores que creas convenientes para realizar los siguientes ejercicios.

Vas a utilizar imágenes de tu sistema como fondo de los elementos. Procura que sean imágenes que dejen ver bien el contenido

### Ejercicio 41:

- Color de fondo del documento: #ccffcc.
- Caja para cualquier <section>
  - o Color de fondo: amarillo
  - o Altura de 300 pixels y anchura de 580 pixels.
  - o Borde de color marrón, con un grosor de 8 pixels, compuesto de guiones.
  - o Relleno:
    - Superior: 10 pixels.

- Derecho: 25 pixels.
- Inferior: 33 pixels.
- Izquierdo: 44 pixels
- Imagen de fondo con repetición. Esta imagen debe estar en la carpeta “imágenes” y puede ser la que tu quieras pero que sea menor que las dimensiones de <section>

**Ejercicio 42:**

- Caja para el primer párrafo perteneciente a la segunda sección
  - Color de fondo blanco
  - Color de texto magenta.
  - Altura de 100 pixels y anchura máxima (100%)
  - Márgenes:
    - Superior: 15 pixels.
    - Derecho: 20 pixels.
    - Inferior : 30 pixels
    - Izquierdo: 55 pixels.
  - Borde de color verde, con un grosor de 2 pixels y continuo.
  - Imagen de fondo sin repetición ajustada al margen superior derecho que ocupe el 40% de ancho por el 65% de alto. Esta imagen debe estar también en la carpeta imágenes.
  - Relleno:
    - Superior: 5 pixels.
    - Derecho: 15 pixels.
    - Inferior: 22 pixels.
    - Izquierdo: 10 pixel

**Ejercicio 43:**

- Caja para el elemento <span> que está dentro del primer párrafo del segundo elemento <section>
  - Color de fondo: gris.
  - Borde rosa de 2 pixels de grosor y compuesto de puntos.

## 4.7. TEXTOS Y FUENTES

La tipografía de un texto se puede cambiar con las numerosas propiedades que ofrece CSS mejorando su presentación y controlando las alineaciones, el interlineado, la separación de palabras, etc. Otra cosa importante en este sentido es que CSS tiene propiedades para manejar también las fuentes o tipología de los caracteres que forman un documento HTML.

### 4.7.1. Propiedades de texto

Modifican y controlan la tipografía de un texto.

Propiedad	Descripción	Valores
<u>color</u>	Define el color del texto	<i>Color</i> , initial, inherit
<u>direction</u>	Especifica la dirección de lectura o escritura del texto	ltr, rtl, initial, inherit
<u>letter-spacing</u>	Incrementa o decrementa el espacio entre los caracteres de un texto	normal, <i>longitud</i> , initial, inherit
<u>line-height</u>	Determina la altura de una línea	normal, <i>número</i> , <i>longitud</i> , <i>número%</i> , initial, inherit
<u>text-align</u>	Especifica la alineación horizontal de una línea	left, right, center, justify, initial, inherit
<u>text-decoration</u>	Especifica la decoración de texto	none, underline, overline, line-through, initial, inherit
<u>text-indent</u>	Especifica la indentación de la primera línea de un bloque de texto	<i>Longitud</i> , <i>número%</i> , initial, inherit
<u>text-shadow</u>	Especifica el sombreado de un texto	h-shadow, v-shadow, <i>blur</i> , <i>color</i> , <i>nombre</i> , initial, inherit
<u>text-transform</u>	Controla la onversión entre mayúsculas y minúsculas de un texto	none, capitalize, uppercase, lowercase, initial, inherit
<u>unicode-bidi</u>	Usado junto con la propiedad <i>direction</i> para permitir la sobreescritura en múltiples lenguajes	normal, embed, bidi-override, initial, inherit
<u>vertical-align</u>	Define la alineación vertical de un elemento	baseline, <i>longitud</i> , <i>número%</i> , sub, super, top, text-top, middle, bottom, text-bottom, initial, inherit
<u>white-space</u>	Especifica cómo manejar los espacios en blanco de un texto	normal, nowrap, pre, pre-line, pre-wrap, initial, inherit

<u>word-spacing</u>	Incrementa o decrementa los espacios en blanco entre palabras	normal, <i>longitud</i> , initial, inherit
---------------------	---------------------------------------------------------------	--------------------------------------------

#### - Alineación

Un texto se alinea con respecto a su elemento contenedor a través de la propiedad **text-align**.

**Ejemplo:**

```

<!DOCTYPE html>
<html>
 <head>
 <title>Texto alineación</title>
 <meta charset="UTF-8">
 <style type="text/css">
 .fondoverde
 {background-color:#ccffcc;}

 section
 {width:430px;
 height:400px;
 background-color:#aaaaff ;}

 .izquierda
 {text-align:left;
 background-color:#99ff99;}

 .derecha
 {text-align:right;
 background-color:orange;}

 .centrado
 {text-align:center;
 background-color:#eeff00; }

 .justificado
 {text-align:justify;
 background-color:yellow;}

 </style>
 </head>
 <body class="fondoverde">
 <h1>Alineaciones de elementos </h1>
 <section>
 <article class="izquierda">
 izquierda izquierda izquierda izquierda
 izquierda izquierda izquierda izquierda
 izquierda izquierda izquierda izquierda
 izquierda izquierda izquierda izquierda
 </article>

 <article class="centrado">
 centrado centrado centrado centrado centrado
 centrado centrado centrado centrado centrado
 centrado centrado centrado centrado centrado
 centrado centrado centrado centrado centrado
 centrado centrado centrado centrado centrado
 </article>

 <article class="derecha">
 derecha derecha derecha derecha derecha
 derecha derecha derecha derecha derecha
 derecha derecha derecha derecha derecha
 derecha derecha derecha derecha derecha
 derecha derecha derecha derecha derecha
 </article>
 </section>
 </body>
</html>

```



```

</article>

<article class="justificado">
 justificado justificado justificado justificado
 justificado justificado justificado justificado
 justificado justificado justificado justificado
 justificado justificado justificado justificado
 justificado justificado justificado justificado
</article>
</section>
</body>
</html>

```



#### - Efectos de texto

Con la propiedad **text-decoration** podemos poner un texto subrayado, tachado o con una línea encima.

### Ejemplo:

```
<!DOCTYPE html>
<html>
 <head>
 <title>Efectos de texto</title>
 <meta charset="UTF-8">
 <style type="text/css">
 .fondoverde
 {background-color:#ccffcc;}

 section
 { width:230px;
 height:100px;
 background-color:orange;}

 .subrayado
 {text-decoration: underline;}

 .sobrelínea
 {text-decoration: overline;}

 .tachado
 {text-decoration:line-through;}

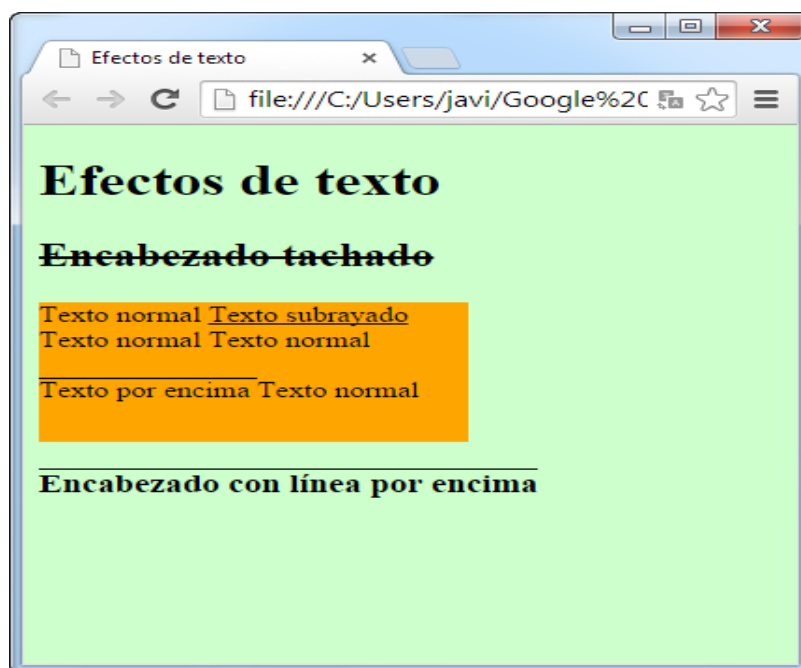
 </style>
 </head>
 <body class="fondoverde">
 <h1> Efectos de texto </h1>
 <h2 class="tachado">Encabezado tachado</h2>
 <section>
 <article>
 Texto normal
 Texto subrayado
 Texto normal Texto normal

 Texto por encima
 Texto normal
```

```

</article>
</section>
<h3 class="sobrelínea">Encabezado con línea por encima</h3>
</body>
</html>

```



En este ejemplo hay que resaltar el uso de del elemento `<span>` para marcar una parte de texto de un bloque y aplicarle un determinado estilo.

#### 4.7.2. Propiedades de las fuentes o tipos de letras

Con estas propiedades podemos manejar la tipología de nuestros textos

Propiedad	Descripción	Valores
font	Sintaxis alternativa	<i>font-style, font-variant, font-weight, font-size/line-height, font-family, caption, icon, menu, message-box, small-caption, status-bar, initial, inherit</i>
font-family	Especifica la fuente del texto	<i>Nombre fuente, initial, inherit</i>
font-size	Especifica el tamaño de la fuente	<i>medium, xx-small, x-small, small, large, x-large, xx-large, smaller, larger, longitud, número%, initial, inherit</i>
font-style	Especifica el estilo de la fuente	<i>normal, italic, oblique, initial, inherit</i>
font-variant	Especifica si la fuente se	<i>normal, small-caps, initial, inherit</i>

	convierte en mayúsculas pequeñas	
<u>font-weight</u>	Especifica el grosor de una fuente	normal, bold, bolder, lighter, <i>número</i> , initial, inherit

**Ejemplo:**

```

<!DOCTYPE html>
<html>
 <head>
 <title>Efectos de tipos de letra</title>
 <meta charset="UTF-8">
 <style type="text/css">
 .fondoverde
 {background-color:#ccffcc;}

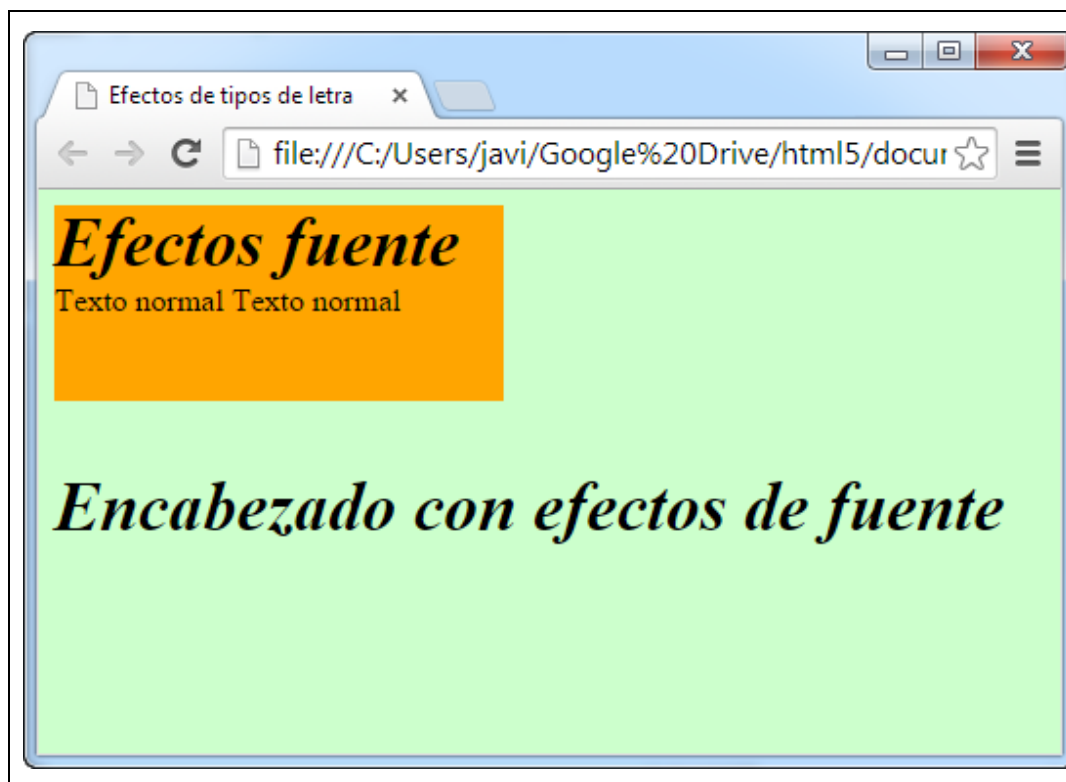
 section
 {width:230px;
 height:100px;
 background-color:orange;}

 .formatofuente
 { font: italic bold 35px Times; }

 </style>
 </head>
 <body class="fondoverde">
 <section>
 <article>
 Efectos fuente

 Texto normal Texto normal
 </article>
 </section>
 <h3 class="formatofuente">Encabezado con efectos de fuente</h3>
 </body>
</html>

```



En este ejemplo el estilo “formatofuente” cambia la fuente de texto de un elemento a Times de 35 pixels cursiva negrita.

### Ejercicios:

Toma de base el siguiente documento html formado por una sección y 4 artículos. Guárdalo como texto.html dentro de la carpeta tema4 y crea la hoja de estilos externa “texto.css” dentro del directorio “estilos”.

```
<!DOCTYPE html>
<html>
 <head>
 <title>Texto alineación</title>
 <meta charset="UTF-8">
 </head>
 <body>
 <h1> Alineaciones de elementos </h1>
 <section>
 <article>
 izquierda izquierda izquierda izquierda
 izquierda izquierda izquierda izquierda
 izquierda izquierda izquierda izquierda
 </article>

 <article>
 centrado centrado centrado centrado centrado
 centrado centrado centrado centrado centrado
 centrado centrado centrado centrado centrado
 </article>

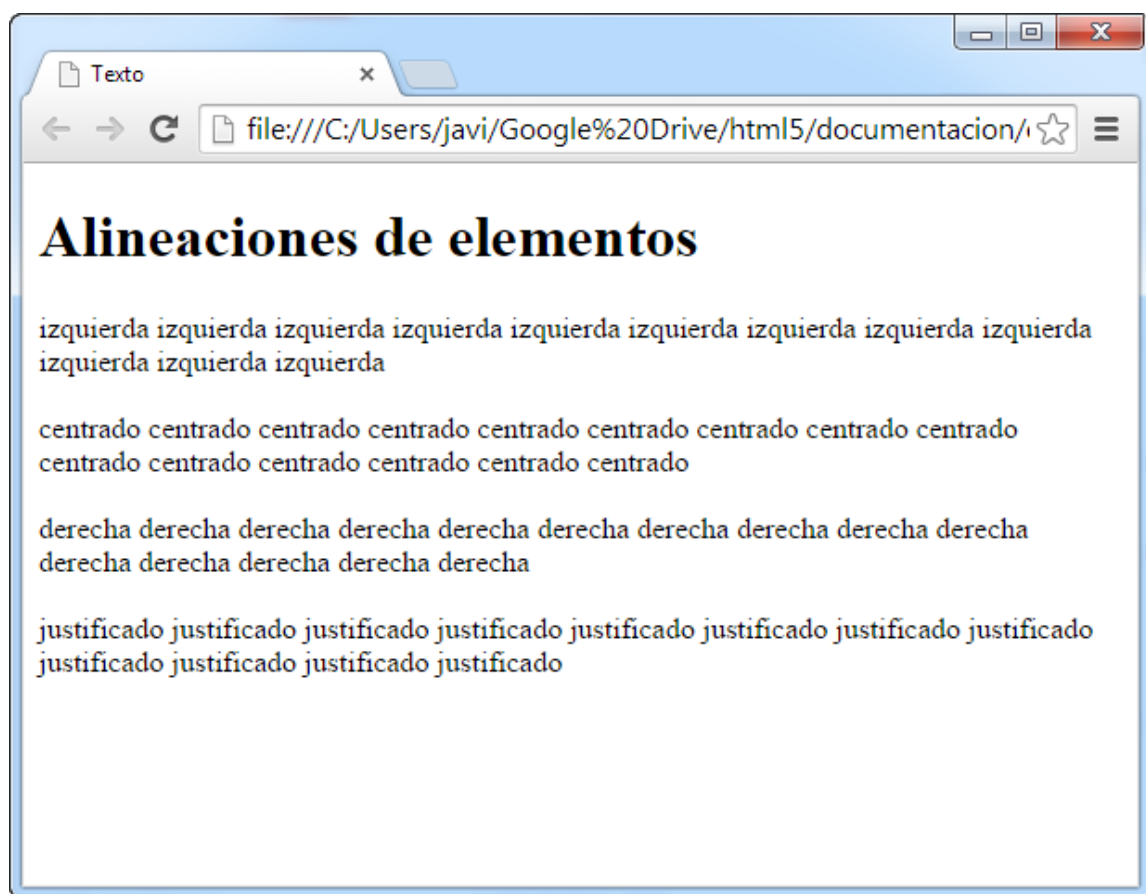
 <article>
```

```

 derecha derecha derecha derecha derecha
 derecha derecha derecha derecha derecha
 derecha derecha derecha derecha derecha
 </article>

 <article>
 justificado justificado justificado justificado
 justificado justificado justificado justificado
 justificado justificado justificado justificado
 </article>
</section>
</body>
</html>

```



#### Ejercicio 44:

- Agrega la hoja de estilos "texto.css" donde vas a ir incorporando los estilos al documento html.
- Agrega la hoja de estilos "cajas.css" para aplicar los estilos que has creado para los elementos <section> y para el color de fondo de <body> al documento html.
- Crea una caja para todos los elementos <article> con:
  - o Color de fondo: gris
  - o Altura de 60 pixels y anchura máxima.
  - o Borde continuo de color azul, con un grosor de 1 pixel
  - o Relleno:

- Superior: 2 pixels.
  - Derecho: 25 pixels.
  - Inferior: 5 pixels.
  - Izquierdo: 34 pixels
- Alinea el texto de cada artículo en base a lo que indica el texto del artículo, es decir, si el texto es “izquierda”, por ejemplo, alinéalo a la izquierda.
  - Los textos de los artículos primero y tercero deben llevar una indentación de 2 pixels, un tamaño de letra de 8 pixels y su tipo de letra será “Times”
  - El texto del segundo artículo estará en negrita, con un tamaño de 10 pixels, subrayada y su tipo de letra será “Arial”
  - El texto del cuarto artículo estará en cursiva y su tipo de letra será “Verdana”



## 4.8. VISIBILIDAD

Tal y como hemos visto en el bloque 2 del curso, los elementos HTML se pueden dividir en dos grupos genéricos, tipo bloque o block y tipo en línea o inline, según la forma de visualización dentro del contexto de una página web.

CSS dispone de propiedades para manipular la forma de visualizar los elementos dentro de un documento HTML. Dos de ellas son:

- **visibility**

Permite hacer que un elemento sea visible o no.

Valor	Descripción
visible	El elemento es visible (opción por defecto)
hidden	El elemento es invisible pero sigue ocupando espacio
collapse	Solo se usa en tablas para eliminar una fila o columna sin afectar a su forma y poder utilizar ese espacio para otro contenido. Si se asigna este valor en otros elementos que no son tablas, éstos se ocultan
initial	Cambia la propiedad a su valor por defecto
inherit	Hereda la propiedad de su elemento padre

Ejemplo:

```
<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>Visible</title>
 <meta charset="UTF-8">
 <style type="text/css">
 .fondoverde
 {background-color:#ccffcc;}

 .oculto
 {visibility: hidden;}
```

```

</style>

</head>

<body class="fondoverde">

 <h1>Encabezado visible</h1>

 <h1 class="oculto">Encabezado invisible</h1>

 <h2>El encabezado superior ocupa espacio pero no se ve</h2>

</body>

</html>

```



En el ejemplo el encabezado `<h1>` con el texto “encabezado oculto” al que se ha aplicado la propiedad *visibility* con el valor *hidden* ocupa su espacio pero no se ve.

- **display**

Con esta propiedad se puede cambiar la forma de visualización de un elemento en un documento HTML.

Valor	Descripción
inline	Visualiza un elemento cómo un elemento en línea (valor

	por defecto)
block	Visualiza un elemento cómo un elemento en bloque
flex	Visualiza un elemento como bloque dentro de un contenedor de “caja flexible”
inline-block	Visualiza un bloque como elemento en línea, es decir, el elemento en sí es en línea aunque internamente el contenido es tipo bloque
inline-flex	Visualiza un contenedor flex como un elemento en línea.
inline-table	El elemento es visualizado en línea aunque internamente se organiza como una tabla
list-item	El elemento se comporta como un componente de una lista
run-in	Visualiza un elemento o bien en línea o bien como bloque, dependiendo del contexto
table	El elemento se comporta como un elemento <table>
table-caption	El elemento se comporta como un elemento <caption>
table-column-group	El elemento se comporta como un elemento <colgroup>
table-header-group	El elemento se comporta como un elemento <thead>
table-footer-group	El elemento se comporta como un elemento <tfoot>
table-row-group	El elemento se comporta como un elemento <tbody>
table-cell	El elemento se comporta como un elemento <td>
table-column	El elemento se comporta como un elemento <col>
table-row	El elemento se comporta como un elemento <tr>
none	El elemento no se visualiza y, además, no ocupa espacio
initial	Cambia la propiedad a su valor por defecto
inherit	Hereda el valor de la propiedad del elemento padre



**display** no cambia la naturaleza del elemento aunque cambie su visualización, es decir, por ejemplo, un elemento tipo in-line se visualiza como bloque pero no puede tener otros elementos dentro de él

### Ejemplo:

En este ejemplo vamos a usar párrafos (elementos de tipo bloque), texto normal y un enlace (elementos inline) para entender cómo funciona la propiedad display para visualizar elementos bloque como inline y viceversa.

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>elementos inline</title>

 <meta charset="UTF-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 p

 { height:50px;

 width:70px;

 background-color:yellow;

 border: 1px solid green;

 }

 p.enlinea

 { display:inline; height:100px;

 width:200px;

 background-color:orange;

 border: 2px dotted black;}
```

```
.bloque
```

```
{display:block; height: 70px;
width:150px;
background:white;
border: 1px dashed brown;
}
```

```
</style>
```

```
</head>
```

```
<body class="fondoverde">
```

```
<p>
```

Párrafo normal

```
</p>
```

Texto en línea normal

```
<p class="enlinea">
```

Párrafo en línea

Párrafo en línea

Párrafo en línea

Párrafo en línea

Párrafo en línea

Párrafo en línea

Párrafo en línea

Párrafo en línea

```
</p>
```

Texto en línea normal <br><br>

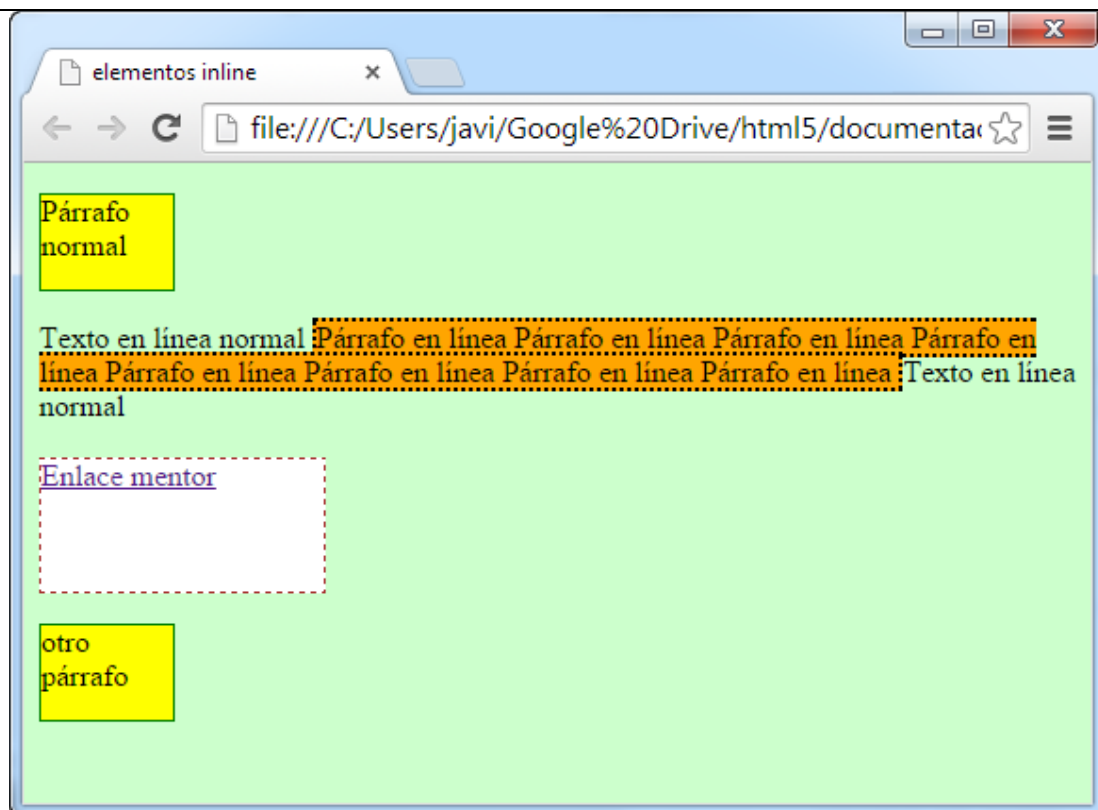
[Enlace mentor](http://www.mentor.mec.es)

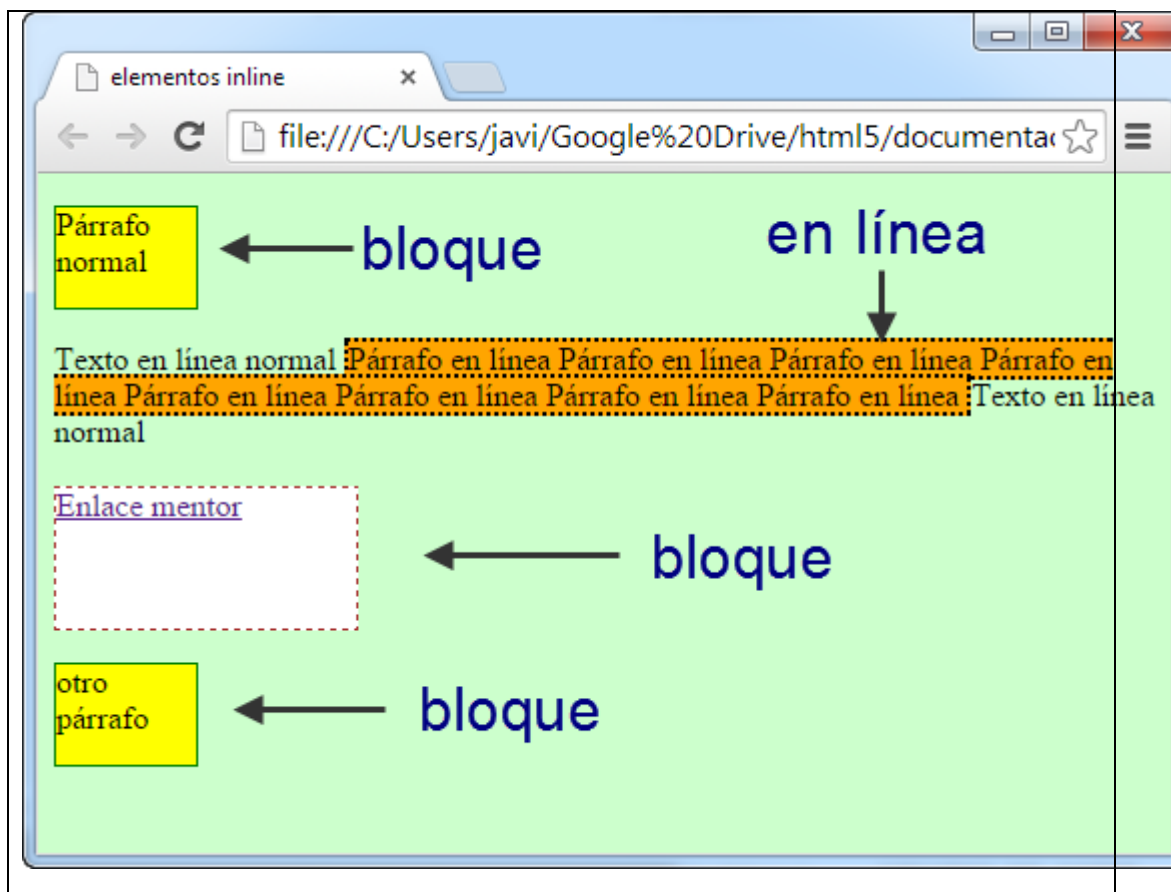
<p>

otro párrafo

&lt;/body&gt;

&lt;/html&gt;





En este ejemplo se usan tres párrafos. Uno de ellos (el que tiene el texto repetido “Párrafo en línea”, borde marrón de puntos de grosor 1px) se visualiza como “inline” y se ve como no actúa como un elemento bloque al no comenzar en una nueva línea y al dejar que se puedan poner texto junto a él (además la propiedad `height` es inútil al visualizar el elemento en línea). En cambio el elemento `<a>` (enlace al proyecto mentor) al que se aplica el estilo “bloque” se comporta como tal, es decir, empieza en una línea nueva y no se pueden poner más elementos a su lado además de tener una altura (`height`), propiedad típica de los elementos de tipo bloque.

### Ejemplo:

Ahora vamos a ver cómo podemos poner un elemento en línea que internamente sea bloque. Para ello vamos a usar el valor *inline-block* de `display` tomando como base el ejemplo anterior.

```
<!DOCTYPE html>

<html lang="es-es">

<head>
```

```
<title>elementos inline</title>
```

```
<meta charset="UTF-8">
```

```
<style type="text/css">
```

```
.fondoverde
```

```
{background-color:#ccffcc;}
```

```
p
```

```
{ height:50px;
```

```
width:70px;
```

```
background-color:yellow;
```

```
border: 1px solid green;
```

```
}
```

```
p.enlineabloque
```

```
{ display:inline-block; height:100px;
```

```
width:200px;
```

```
background-color:orange;
```

```
border: 2px dotted black;}
```

```
.bloque
```

```
{display:block; height: 70px;
```

```
width:150px;
```

```
background:white;
```

```
border: 1px dashed brown;
```

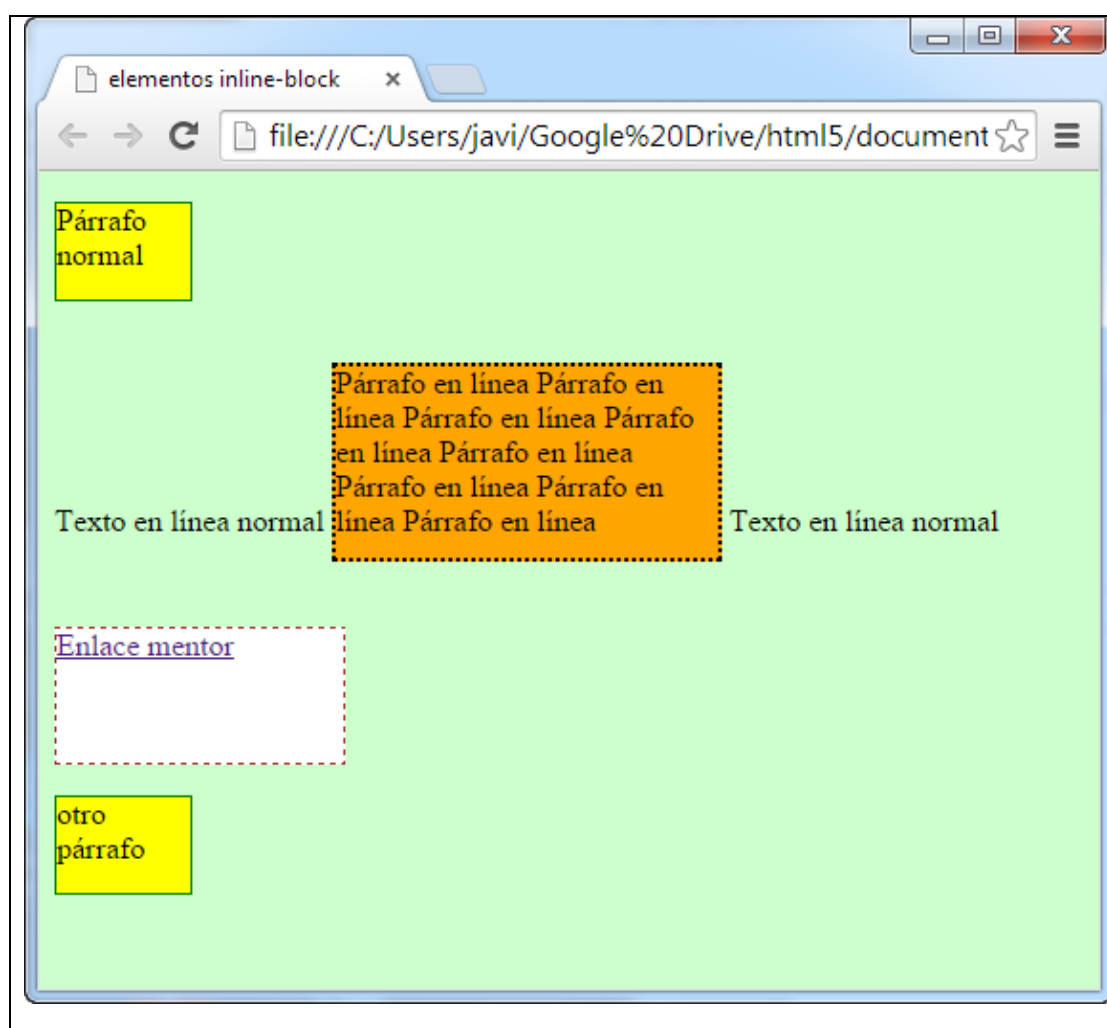
```
}
```

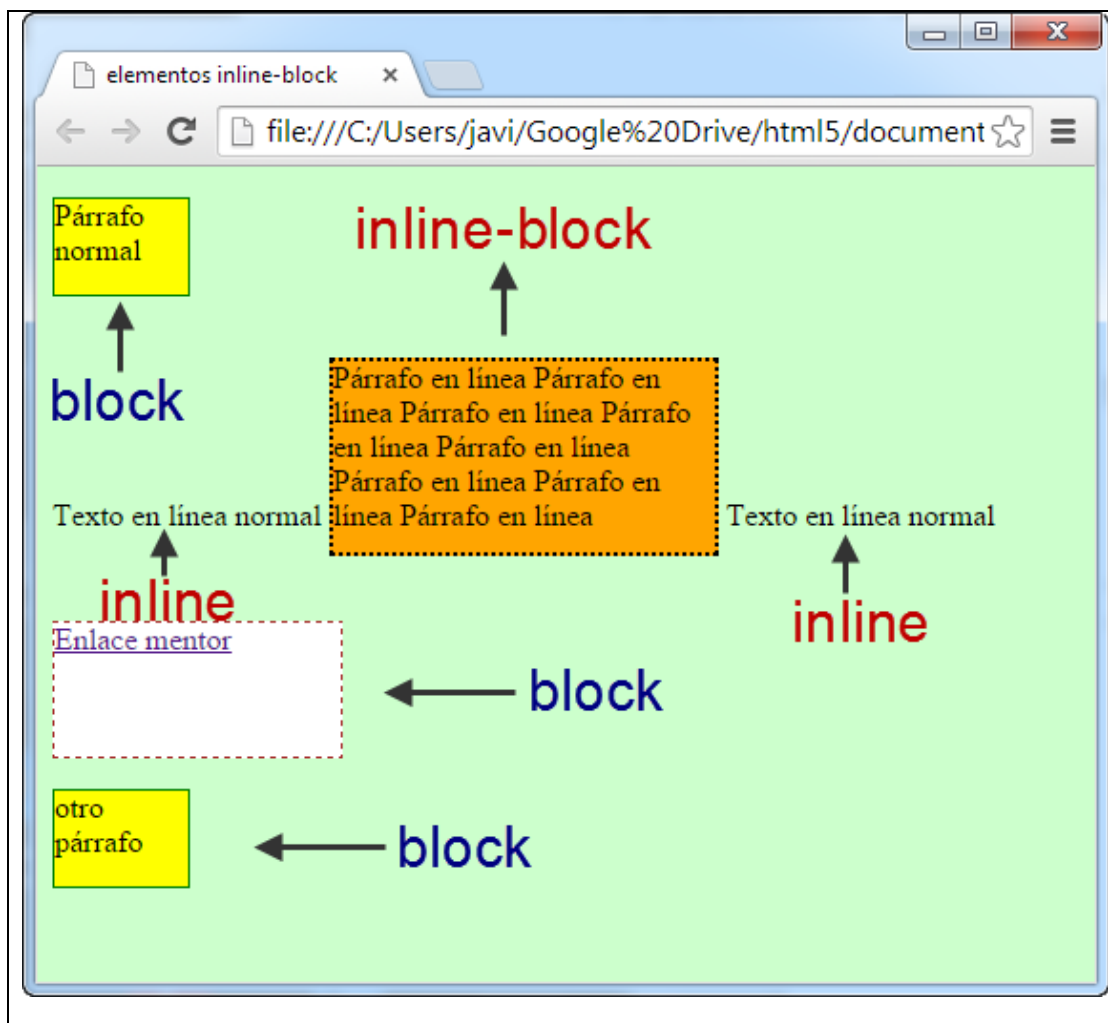


```
</style>
</head>
<body class="fondoverde">

 <p>
 Párrafo normal
 </p>
 Texto en línea normal
 <p class="enlineabloque">
 Párrafo en línea
 Párrafo en línea
 Párrafo en línea
 Párrafo en línea
 Párrafo en línea
 Párrafo en línea
 Párrafo en línea
 Párrafo en línea
 </p>
 Texto en línea normal

 Enlace mentor
 <p>
 otro párrafo
 </p>
</body>
</html>
```





En este ejemplo el párrafo anterior que teníamos en línea ahora es inline-block, es decir, se ve en línea (se alinea el texto con la última línea del párrafo) pero su interior es bloque (fijarse en que el parámetro height funciona dándole una altura determinada)

Ahora vamos a ver alguna de las maneras que tenemos de visualizar elementos de tipo bloque para ponerlos a la misma altura

### Ejemplo:

Pongamos tres elementos <article> a la misma altura usando el valor inline-block de display.

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>elementos table</title>
```

```
<meta charset="UTF-8">

<style type="text/css">

.fondoverde

 {background-color:#ccffcc;}

 article

 {

 display:inline-block;

 height: 150px;

 background-color:yellow;

 border: 2px dotted black;

 }

</style>

</head>

<body class="fondoverde">

 <article>

 Texto del primer artículo

 </article>

 <article>

 Texto del segundo artículo

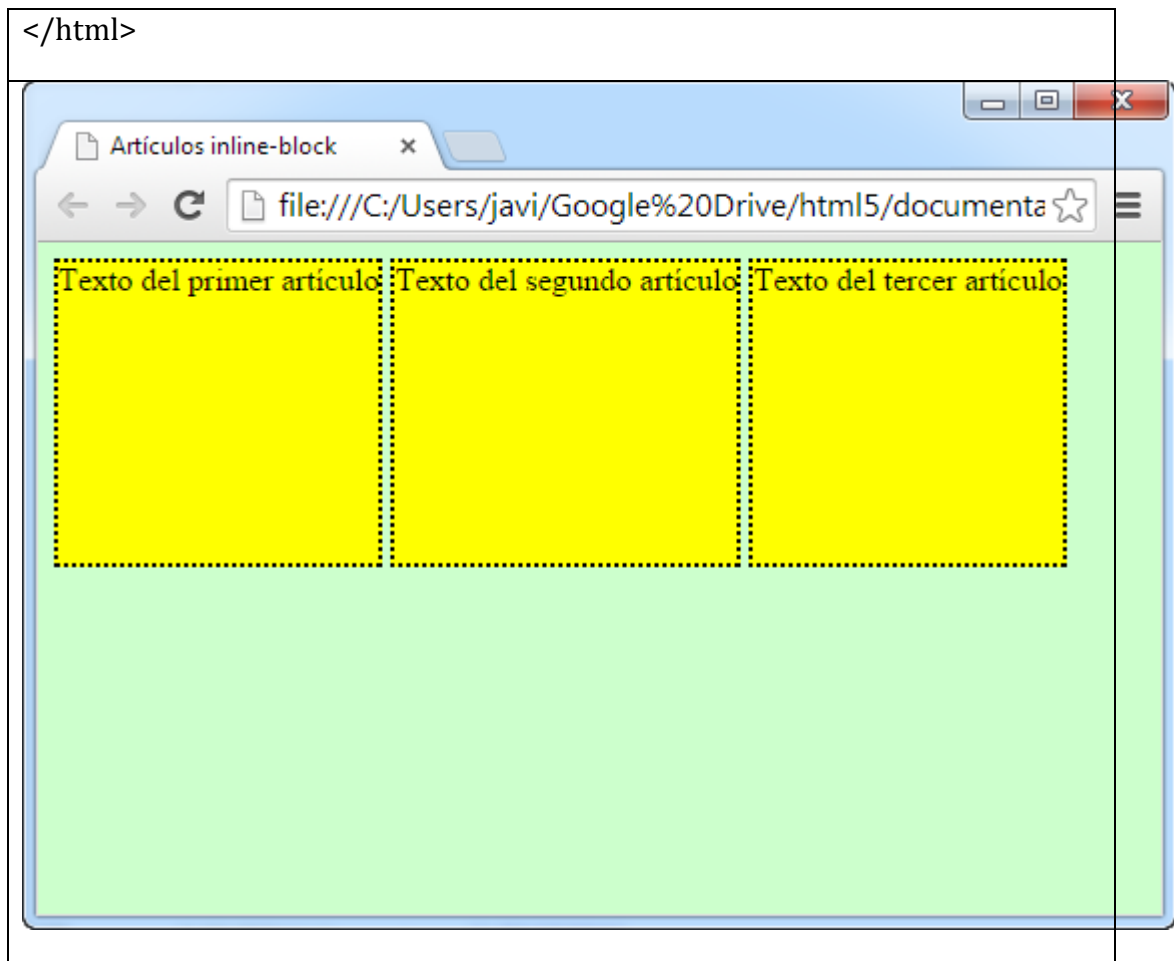
 </article>

 <article>

 Texto del tercer artículo

 </article>

</body>
```



El gran problema que tiene esta forma de ajustar elementos tipo bloque es que si cambiamos el tamaño de la ventana, la estructura que hemos creado se desorganiza como vemos en la imagen inferior



### Ejemplo

Otra forma que tenemos de visualizar agrupaciones en línea de elementos tipo bloque es la tabular o la visualización de dichos elementos como si fueran una tabla o un componente de la misma.

Vamos a usar el valor `table-cell` de `display`, es decir, que los elementos se comporten como si fuera celdas de una tabla (recuerda que las celdas de las tablas siempre se construyen por filas, es decir, una junta a otra o en línea).

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Artículos table-cell</title>

 <meta charset="UTF-8">

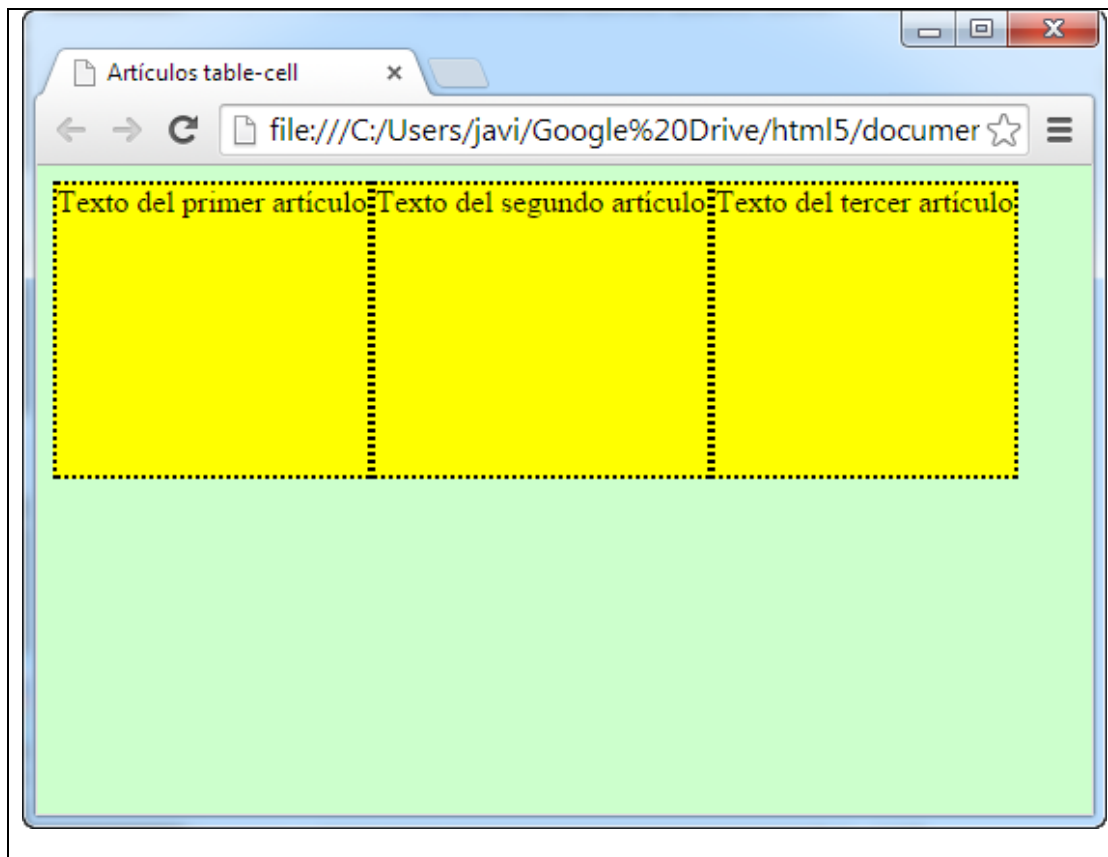
 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}
```

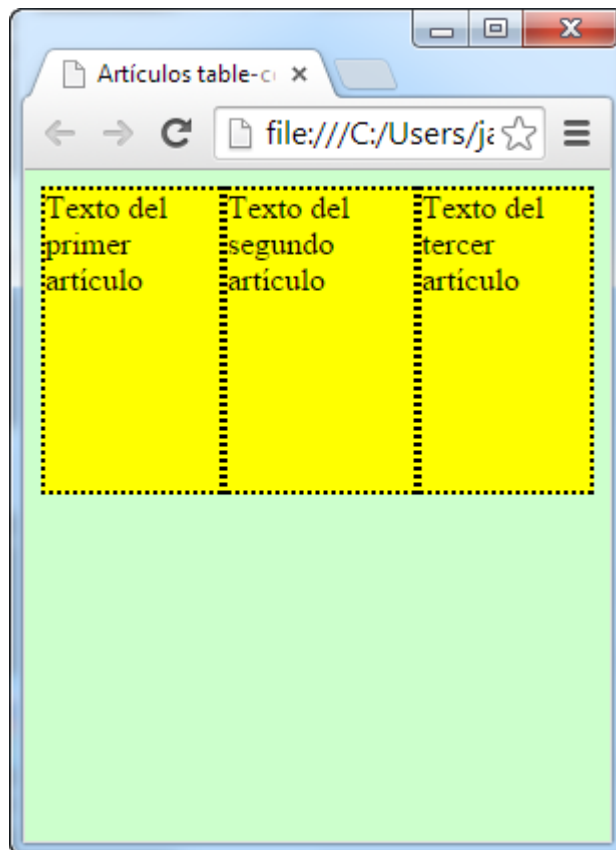
```
 article
 {
 display:table-cell;
 height: 150px;
 background-color:yellow;
 border: 2px dotted black;
 }

</style>
</head>
<body class="fondoverde">
 <article>
 Texto del primer artículo
 </article>
 <article>
 Texto del segundo artículo
 </article>
 <article>
 Texto del tercer artículo
 </article>
</body>
</html>
```



Esta forma evita la desorganización de elementos que ocurría con la visualización inline-block tal y como vemos en la figura de abajo, pero nos obliga a usar los elementos como si fueran celdas de una tabla en cuanto a márgenes, espacios entre el contenido y los márgenes, etc.

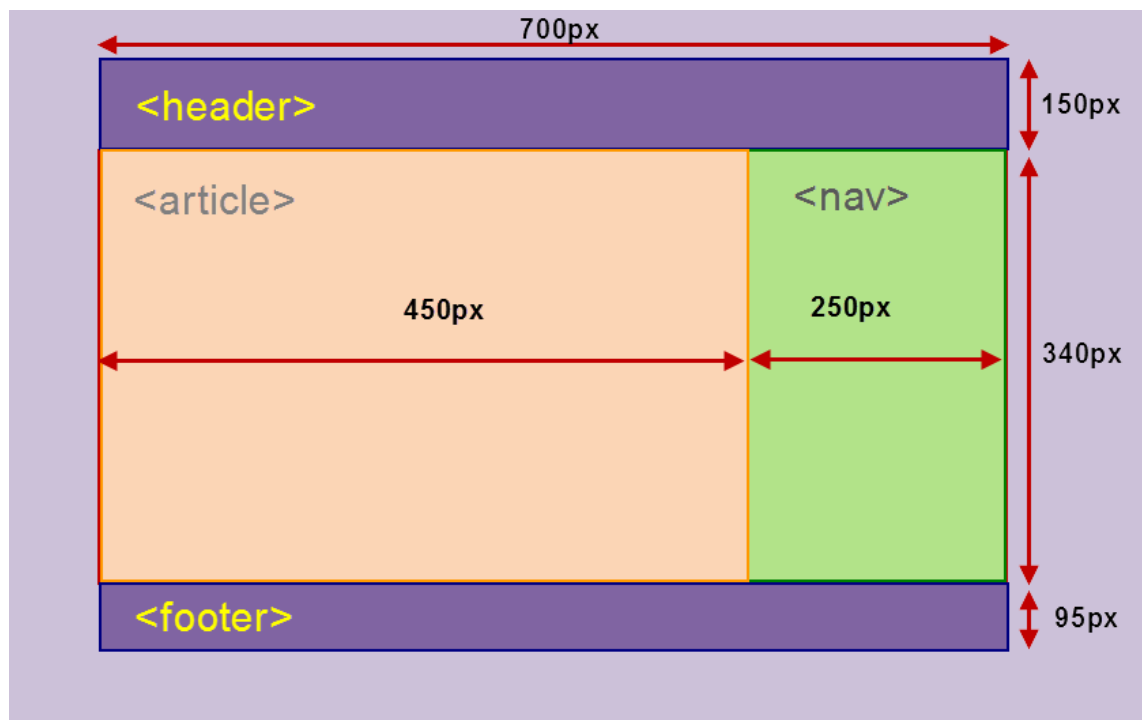




### Ejercicios:

#### Ejercicio 45:

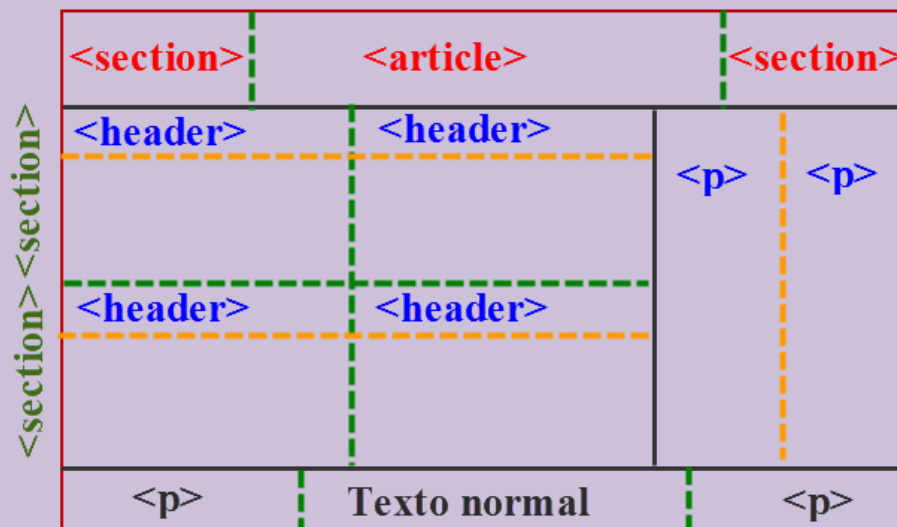
Poniendo los colores de fondo que quieras crea la siguiente página simple con esta estructura y el texto que quieras, guárdalo como ejercicio45.html en la carpeta tema4, la hoja de estilos que crees llámala "ejercicio45.css" y guárdala en el directorio estilos.



#### Ejercicio 46:

Tomando como base el ejercicio anterior vamos a hacer una variación un poco más compleja dividiendo cada uno de los elementos anteriores y creando otros dentro de ellos tal y como se te indica a continuación

- `<header>`
  - `<section>`: ancho 20%
  - `<article>`: ancho 60%
  - `<section>`: ancho 20%
- `<article>`
  - Tiene cuatro `<section>` iguales formando un cuadrado
  - `<section>`: ancho 50% y alto 50%
    - `<header>`: 30 pixels de alto
    - Texto normal
- `<nav>`
  - Dos elementos `<p>` con 50% de ancho
- `<footer>`
  - `<p>`: ancho 33%
  - Texto normal que ocupe en total un ancho de 33%
  - `<p>`: ancho 34%



Guarda el ejercicio como ejercicio46.html y la hoja de estilos correspondiente como ejercicio46.css

## 4.9. POSICIONAMIENTO

Cada caja que se forma cuando se inserta un elemento en un documento HTML ocupa un espacio y tiene una localización determinada por los siguientes factores:

- Si el elemento HTML es de tipo bloque o en línea
- Tipo de posicionamiento de la caja (normal, relativo, absoluto, fijo o flotante)
- La interrelación entre los elementos, es decir, si un elemento está definido dentro de otro, si se le aplica herencia de su elemento padre, etc.
- La modificación de las dimensiones de la caja a través de las propiedades `width` y `height`.
- Adaptación al medio de visualización de la dimensión de las cajas para su mejor visualización.
- Ajuste de las dimensiones de la caja al contenido del elemento como ocurre con tablas o imágenes.

A través de CSS y su propiedad **position** podemos definir y manipular la posición de la caja de un determinado elemento a excepción del posicionamiento flotante que se debe manejar a través de la propiedad **float**.

Los posibles valores de **position** son

Valores	Descripción
static	Posicionamiento normal o estático (por defecto)
absolute	Posicionamiento absoluto
fixed	Posicionamiento fijo
relative	Posicionamiento relativo
initial	Asigna a la propiedad su valor por defecto
inherit	Asigna el valor de la propiedad del elemento padre

Los principales propiedades en las que se apoya **position** para calcular la posición de un elemento son:

Propiedades	Descripción
top	Distancia al borde superior del contenedor o desplazamiento vertical hacia abajo
right	Distancia al borde derecho del contenedor o desplazamiento horizontal hacia la izquierda

bottom	Distancia al borde inferior del contenedor o desplazamiento vertical hacia arriba
left	Distancia al borde izquierdo del contenedor o desplazamiento horizontal hacia la derecha
z-index	Indica el orden de superposición de un elemento con respecto a otros

#### 4.9.1 Posicionamiento normal o estático

Es el que emplean por defecto los navegadores para mostrar en una posición la caja de un elemento concreto. El cálculo de esta posición depende de si el elemento es de tipo bloque o en línea, de su contenido y de las propiedades width y height de dicho elemento y del flujo de la página.

	<i>Las propiedades CSS top, bottom, left y right no son aplicables en el posicionamiento estático</i>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------

#### 4.9.2. Posicionamiento relativo

Es aquel en el que se desplaza la caja de un elemento a una posición determinada por la posición normal de la misma a la que se aplica un desplazamiento horizontal, vertical en ambos sentidos o combinación de los anteriores a través de las propiedades CSS top, left, bottom y right.

	<i>Las propiedades top, left, bottom y right se le pueden asignar valores numéricos negativos lo que implica que el desplazamiento es en sentido contrario a la naturaleza de la propiedad, por ejemplo, si top vale -5px el desplazamiento es hacia arriba en vez de hacia abajo</i>
-------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<i>Nunca emplees las propiedades left y right o top y bottom juntas ya que son excluyentes entre si y uno de los componentes de las parejas anteriores no funcionará</i>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### Ejemplo:

Vamos a usar un elemento <article> para desplazarlo 50 pixels hacia abajo y 100 pixels a la derecha desde la posición inicial o normal en la que estaría el elemento al añadirlo al documento HTML.

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Posicionamiento relativo</title>

 <meta charset="UTF-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 article.normal

 {

 width:250px;

 height:100px;

 background-color: orange;

 }

 article.desplazado

 {

 position: relative;

 top:50px;

 left:100px;

 }

 </style>

</head>

<body class="fondoverde">

 < article class="normal desplazado">

 Texto del artículo
```

Texto del artículo

Texto del artículo

Texto del artículo

Texto del artículo

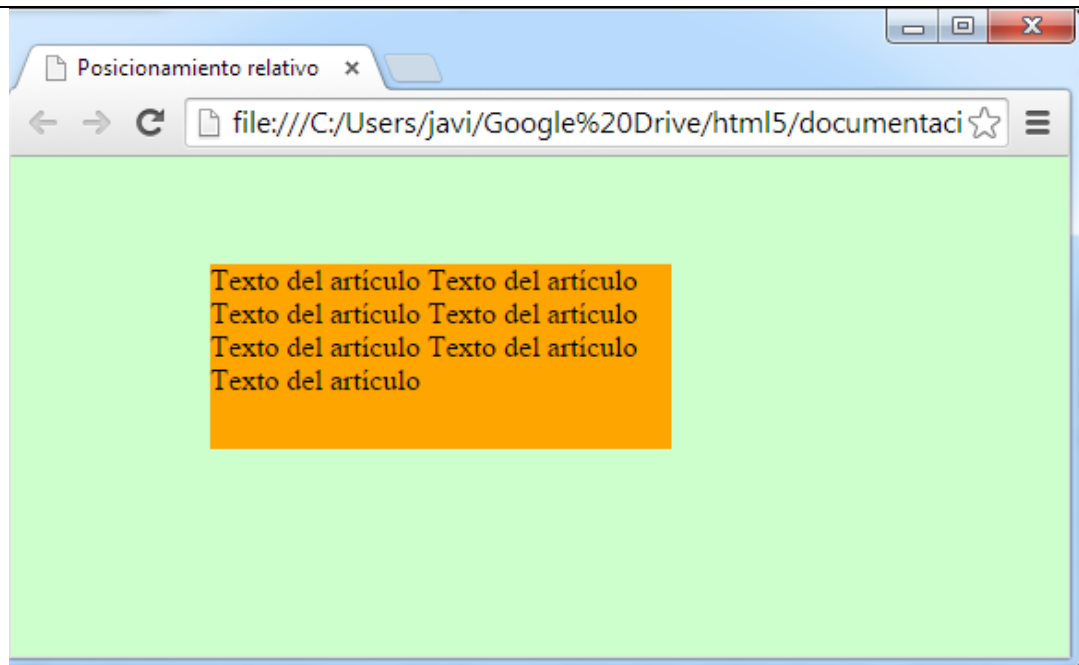
Texto del artículo

Texto del artículo

`</article>`

`</body>`

`</html>`





En este ejemplo el elemento `<article>` es el primero que ponemos en nuestro documento HTML por lo que su posición normal sería la esquina superior izquierda de su contenedor (en este caso `<BODY>`). Aplicamos dos estilos a `<article>` una para el aspecto (class "normal") y otro para el "movimiento" (class "desplazado") que hace que el elemento se desplace 50 pixels del borde superior de `<BODY>` y 100 pixels del borde izquierdo también de `<BODY>`.

### Ejemplo:

Ahora vamos a ver una característica muy importante que debemos saber y es que cualquier tipo de desplazamiento siempre va referido al elemento contenedor. Para ello vamos a definir un artículo normal, luego una sección desplazada 100 pixels desde el borde superior y 100 pixels desde el borde izquierdo que, a su vez, contenga un artículo desplazado 50 pixels desde el borde superior y 100 pixels desde el borde izquierdo (usaremos el ejemplo anterior como base)

```
!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Posicionamiento relativo sección</title>

 <meta charset="UTF-8">

 <style type="text/css">
```



```
.fondoverde
{background-color:#ccffcc;}

#contenedor
{
 position:relative;
 top:100px;
 left:100px;
 width:400px;
 height:200px;
 background-color:grey;
}

article.normal
{
 width:250px;
 height:100px;
 background-color: orange;
}

article.desplazado
{
 position: relative;
 top:50px;
 left:100px;
}

</style>
```

```
</head>

<body class="fondoverde">

 <article class="normal">

 Texto del artículo

 Texto del artículo

 Texto del artículo

 Texto del artículo

 Texto del artículo

 Texto del artículo

 Texto del artículo

 </article>

 <section id="contenedor">

 <article class="normal desplazado">

 Texto del artículo

 Texto del artículo

 Texto del artículo

 Texto del artículo

 Texto del artículo

 Texto del artículo

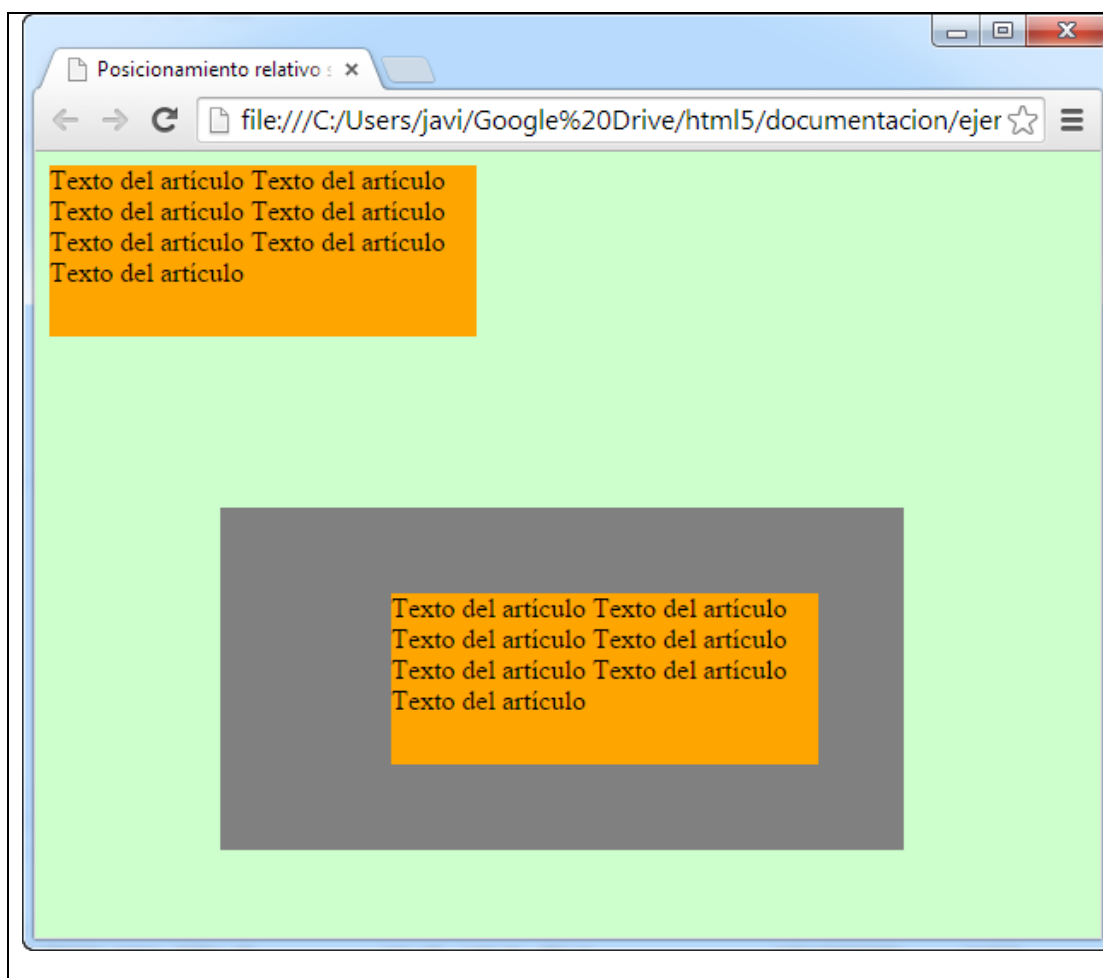
 Texto del artículo

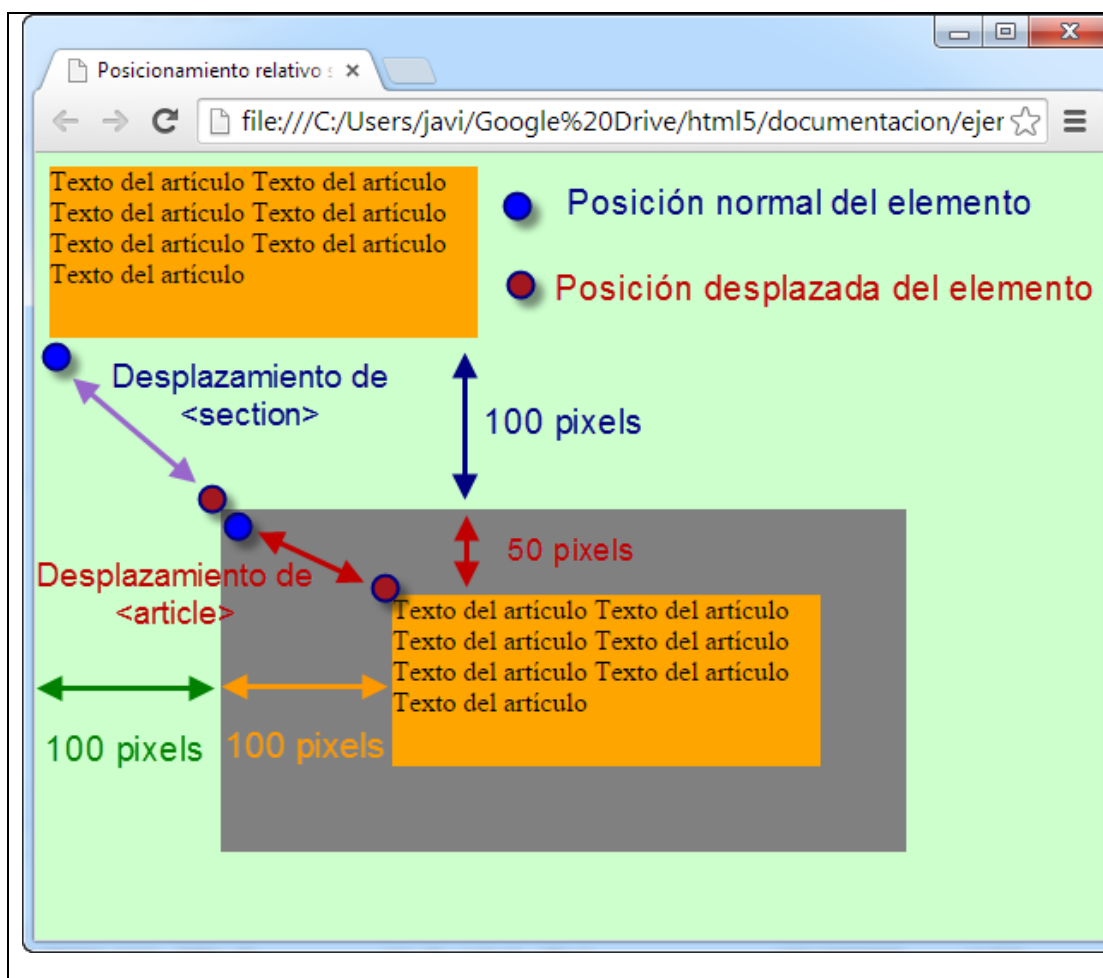
 </article>

 </section>

</body>

</html>
```





En este ejemplo se ve que el elemento `<section>` tiene la su posición normal justo debajo del elemento `<article>` y, desde ahí, se desplaza 100 pixels hacia abajo y 100 pixels a la izquierda. Ahora, al crear otro elemento `<article>` dentro de `<section>`, éste tiene su posición normal en la esquina superior izquierda de `<section>` y para el desplazamiento se toma ese punto como base (50 pixels hacia abajo y 100 pixels a la derecha).

#### 4.9.3. Posicionamiento absoluto

Establece la posición exacta de la caja de un elemento a través de las propiedades `top`, `left`, `right` y `bottom` sin tener en cuenta su posición normal dentro de un elemento contenedor.

##### Ejemplo:

Vamos a poner un párrafo con posicionamiento estático y un encabezado `<h2>` con posicionamiento absoluto.

```
<!DOCTYPE html>

<html lang="es-es">
```

```
<head>

 <title>Posicionamiento absoluto</title>

 <meta charset="UTF-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 article.normal

 {

 width:250px;

 height:100px;

 background-color: orange;

 }

 h2

 {

 position:absolute;

 top:150px;

 left:130px;

 }

 </style>
</head>

<body class="fondoverde">

 <article class="normal">

 Texto del artículo

 Texto del artículo
```

Texto del artículo

Texto del artículo

Texto del artículo

Texto del artículo

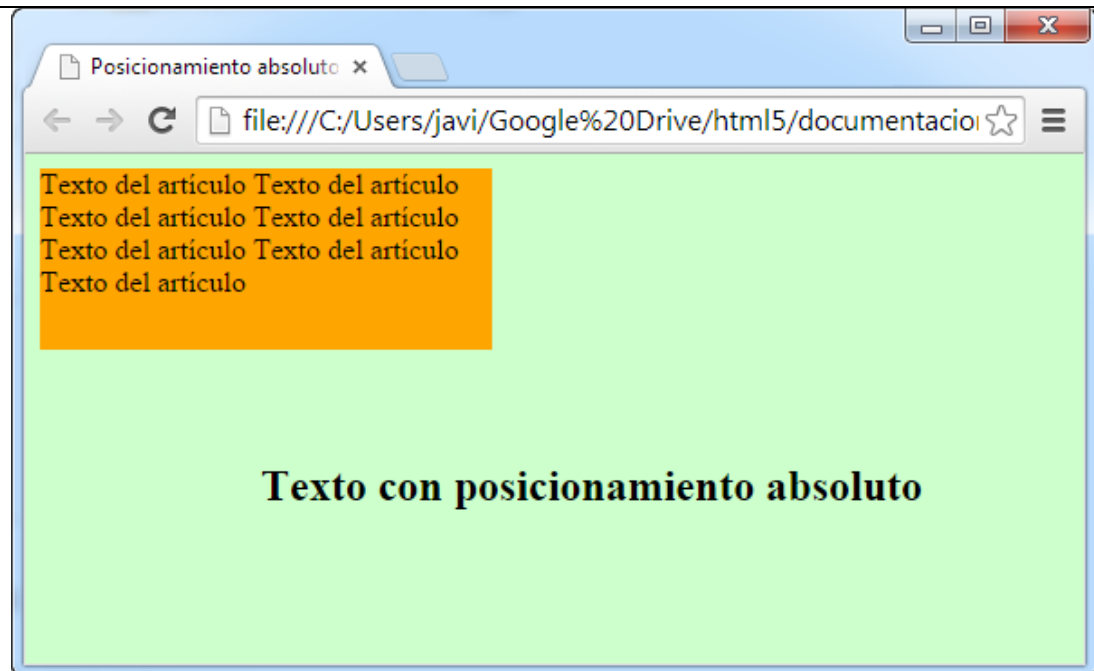
Texto del artículo

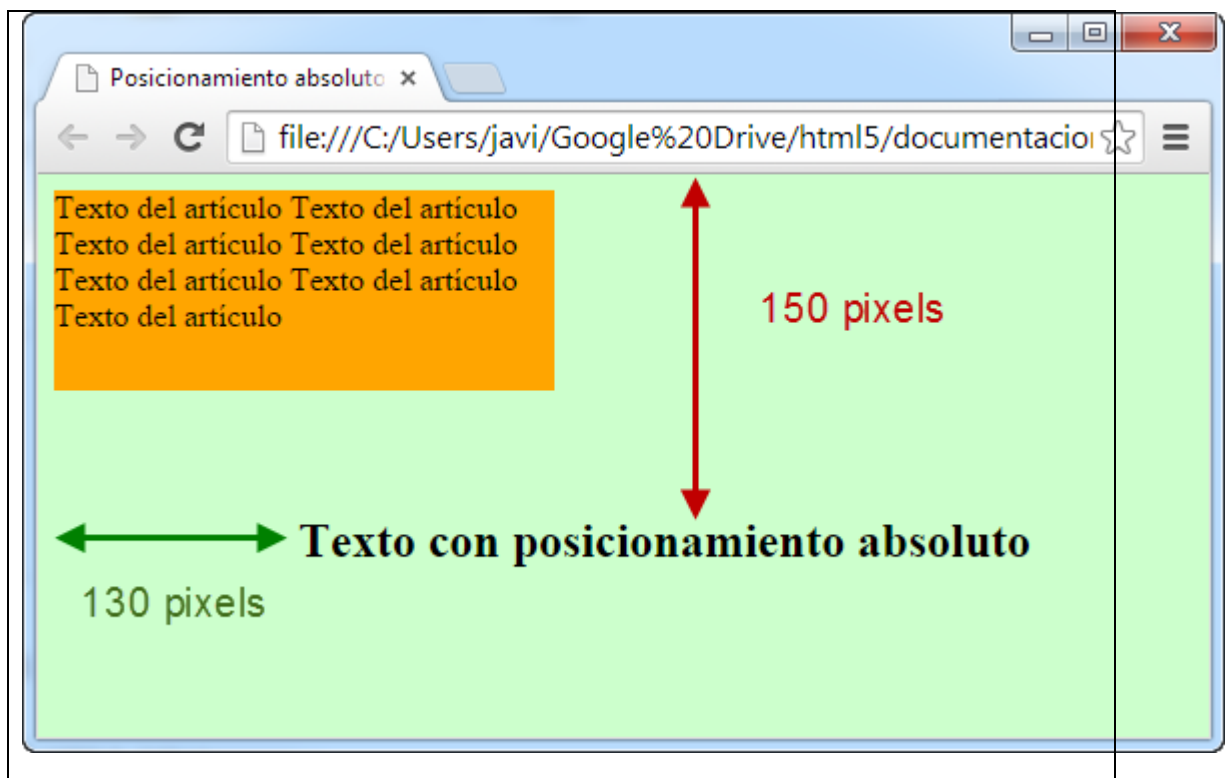
</article>

<h2>Texto con posicionamiento absoluto</h2>

</body>

</html>





Como vemos en el ejemplo el encabezado `<h2>` se coloca a 150 pixels del borde superior de `<BODY>` y a 130 pixels del borde izquierdo también de `<BODY>` independientemente de la posición que ocupe el párrafo que hemos puesto. Esto significa que el posicionamiento absoluto no depende para nada de la posición normal en la que debería ir el elemento y toma como referencia los bordes del contenedor.

### Ejemplo:

En este ejemplo vamos a ver el posicionamiento absoluto dentro de un elemento contenedor como es `<section>`

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Posicionamiento absoluto</title>

 <meta charset="UTF-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 </style>

</head>

<body>

 <div>

 <h2>Posicionamiento absoluto</h2>

 <p>Texto del artículo Texto del artículo</p>
 <p>Texto del artículo Texto del artículo</p>
 <p>Texto del artículo Texto del artículo</p>
 <p>Texto del artículo</p>

 </div>

</body>

</html>
```

```
#contenedor
{
 position:relative;
 top:100px;
 left:100px;
 width:400px;
 height:250px;
 background-color:grey;
}

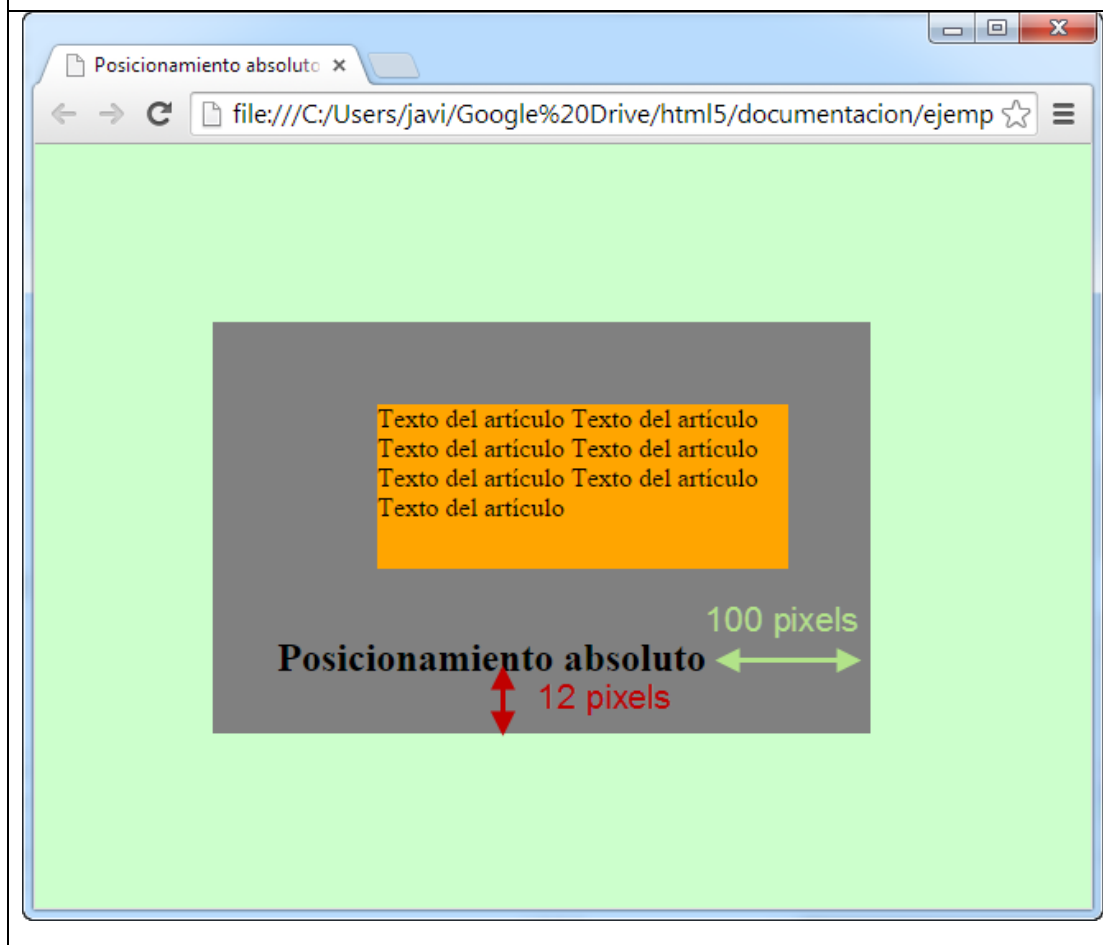
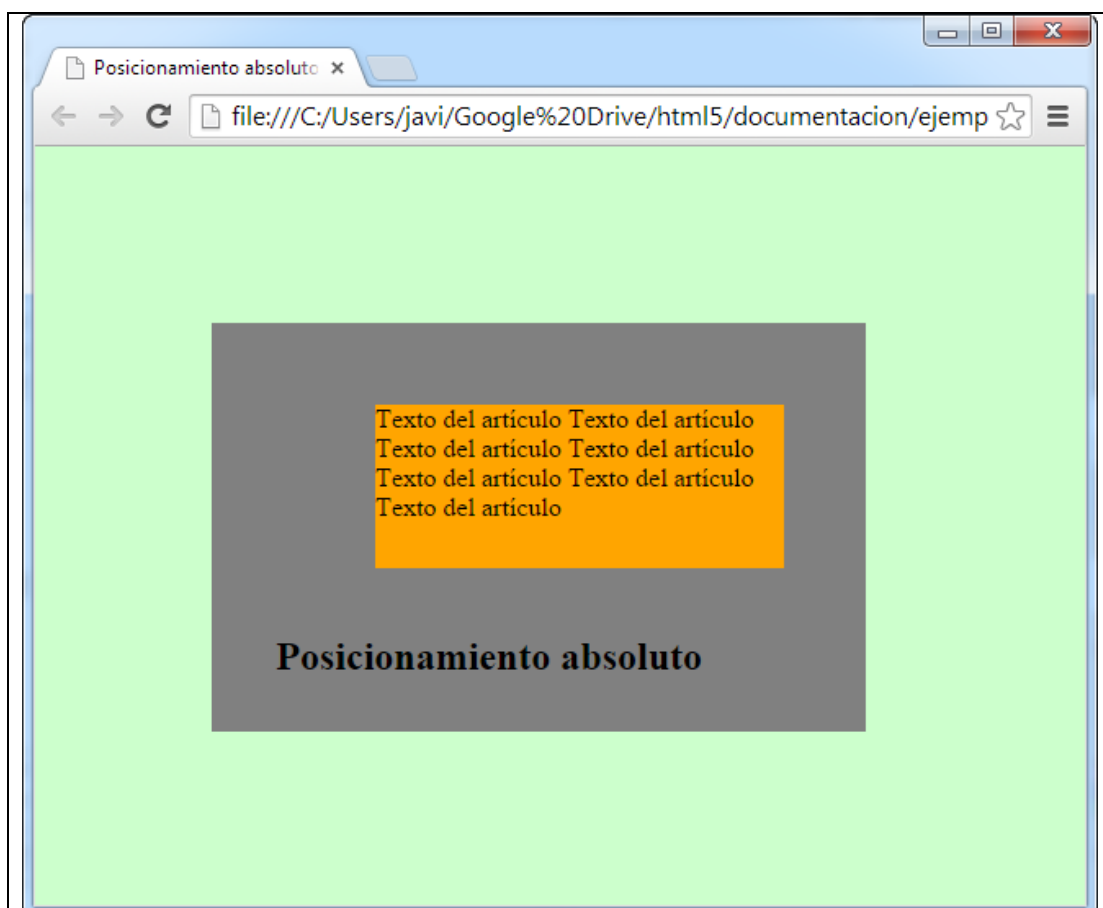
article.normal
{
 width:250px;
 height:100px;
 background-color: orange;
}

article.desplazado
{
 position: relative;
 top:50px;
 left:100px;
}

h2
{
 position:absolute;
```



```
 bottom:12px;
 right:100px;
 }
</style>
</head>
<body class="fondoverde">
<section id="contenedor">
 <article class="normal desplazado">
 Texto del artículo
 Texto del artículo
 Texto del artículo
 Texto del artículo
 Texto del artículo
 Texto del artículo
 Texto del artículo
 </article>
 <h2>Posicionamiento absoluto</h2>
</section>
</body>
</html>
```



Como se ve en el ejemplo la posición del elemento <h2> no depende del párrafo que hay en la sección y su posición está en función de los bordes del elemento <section> que lo contiene. La propiedad bottom nos permite “subir” el elemento desde el borde inferior y la propiedad right nos permite desplazar hacia la izquierda dicho elemento.

#### 4.9.4. Posicionamiento fijo

Es muy similar al posicionamiento absoluto ya que se calcula la posición de la caja del elemento de la misma manera pero, mientras en el posicionamiento fijo dicha caja es inamovible (si se hace scroll de la página, la caja del elemento sigue manteniendo la misma posición en el navegador aunque el resto de los elementos se muevan), en el posicionamiento absoluto la caja del elemento se mueve con los demás elementos de la página manteniendo la posición con respecto a ellos.

##### Ejemplo:

```
<!DOCTYPE html>

<html>

<head>

 <title>Posicionamiento fijo</title>

 <meta charset="UTF-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 .fijo

 { position:fixed;

 top:100px;

 left:50px;

 }
```

```
.dimensión
{
 width:150px;
 height:100px;
}
</style>
</head>
<body class="fondoverde">

 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1

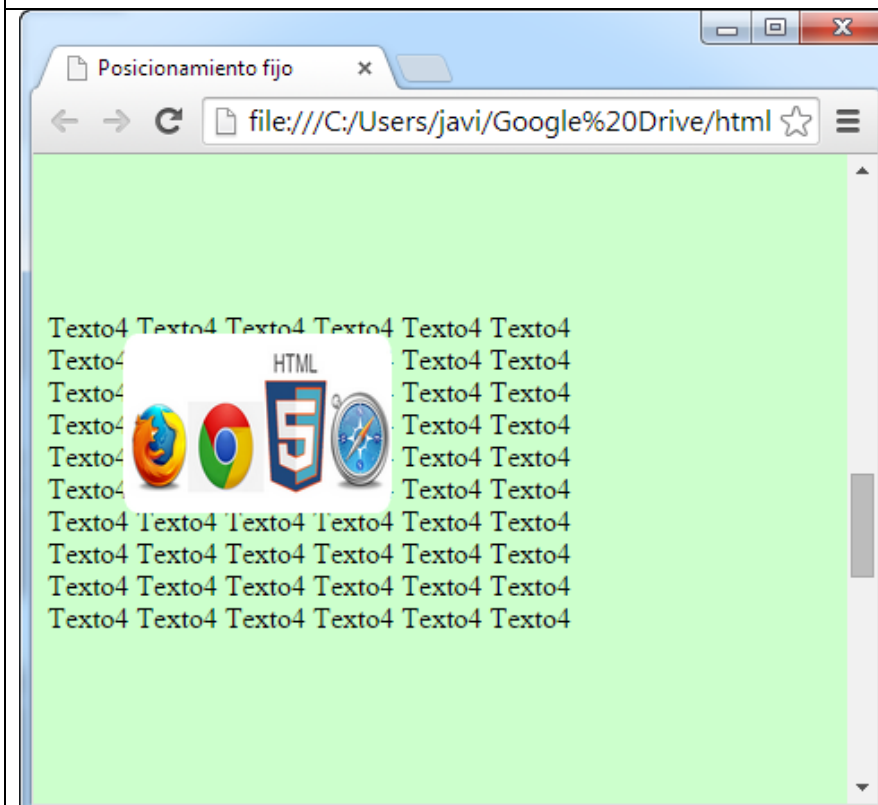
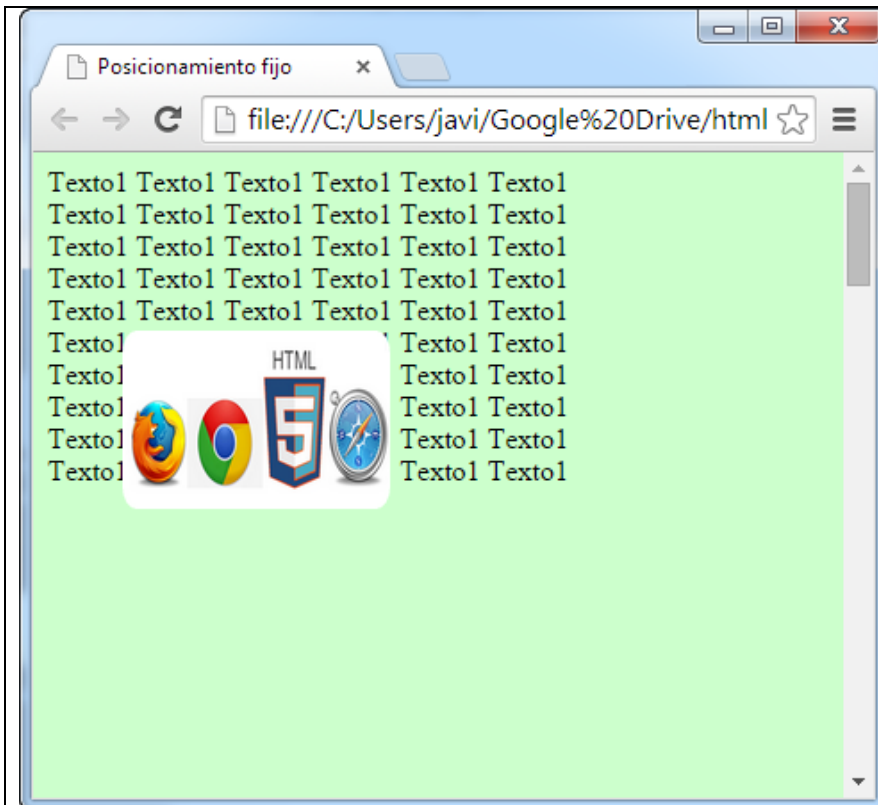
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1

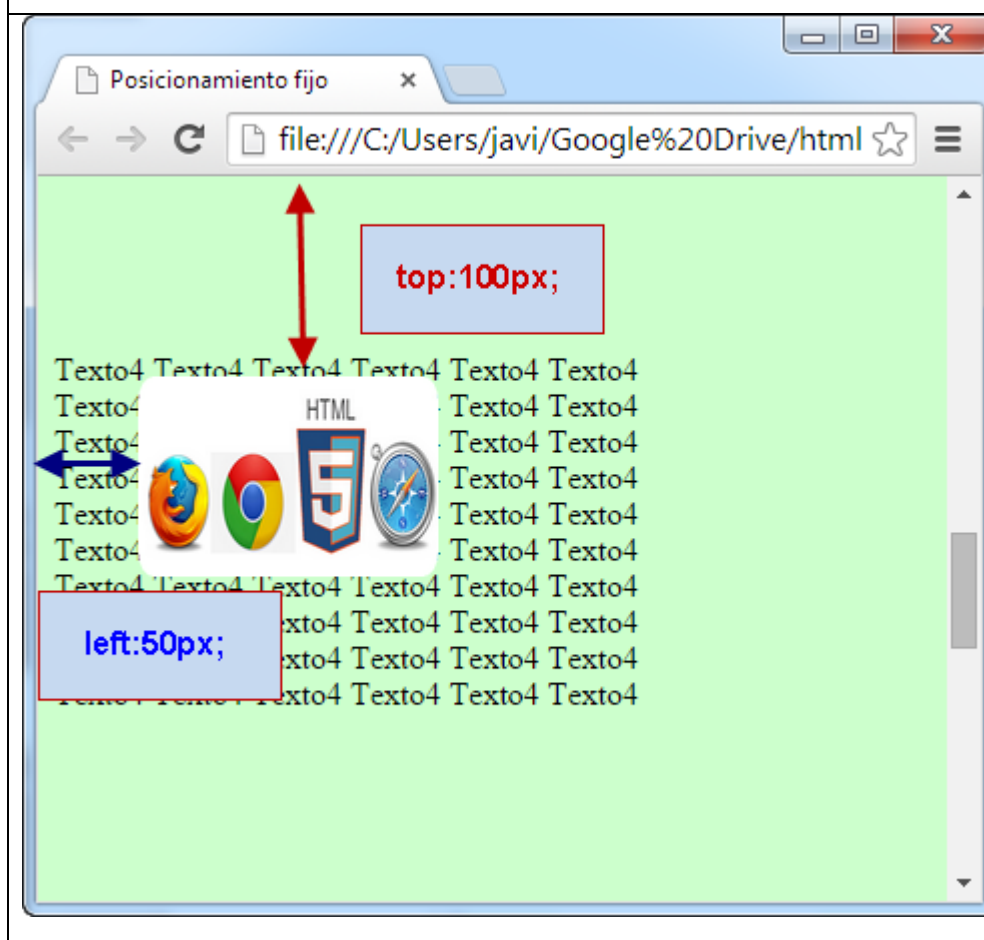
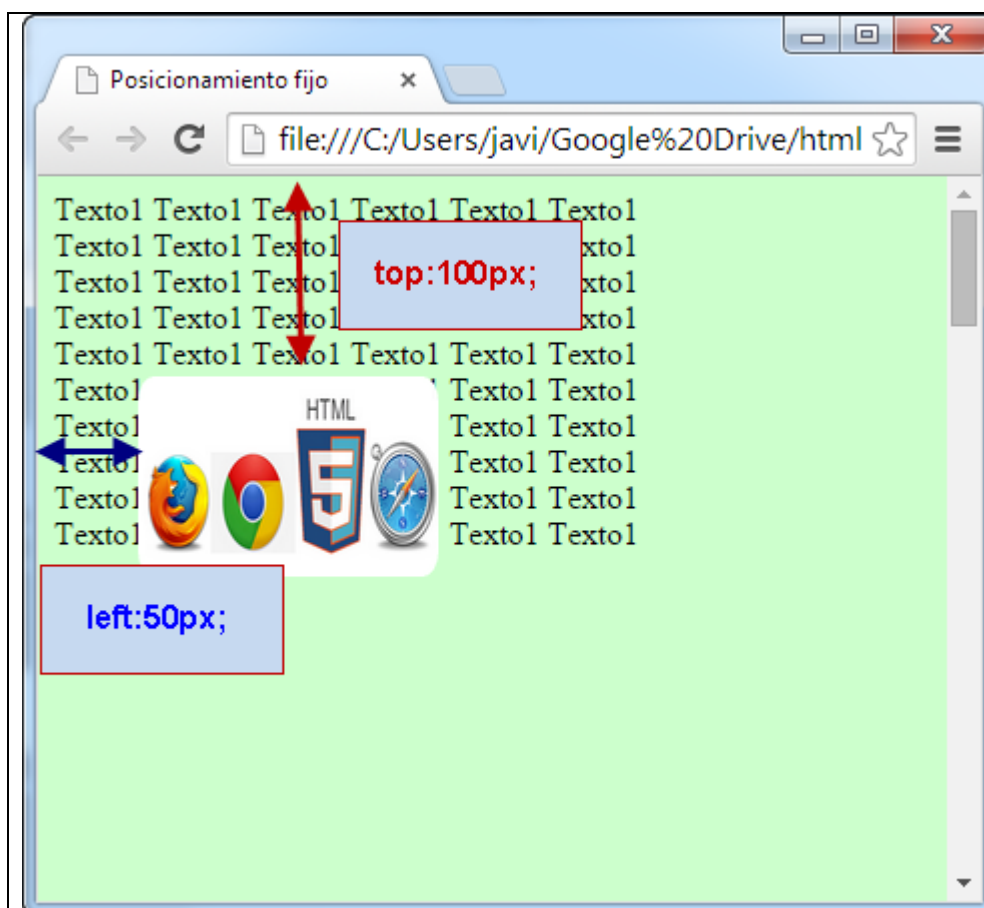
 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1

 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1

 Texto1 Texto1 Texto1 Texto1 Texto1 Texto1

</body>
</html>
```





Para este ejemplo tomamos como base el archivo ejercicio19.html, que empleamos para los enlaces internos, donde colocamos una imagen en una determinada posición (100 pixels del borde superior y 50 pixels del borde izquierdo) con posicionamiento fijo a través de la clase “fijo” y le damos un determinado ancho y alto con la clase “dimensión”. La primera de las figuras sería una captura cuando el navegador carga por primera vez la página (fíjate en el botón de scroll de la derecha) y la segunda de las figuras sería cuando hacemos scroll (fíjate donde está el botón anterior) y como la imagen sigue manteniendo la posición que tenía en la primera figura a pesar de que el texto de la página ha cambiado al hacer scrolling.

#### 4.9.5. Solapamiento de elementos

Este posicionamiento permite visualizar las cajas de los elementos que se solapan en un determinado orden, es decir, indicando cual se verá por encima de las demás.

Se controla con la propiedad **z-index**

Valor	Descripción
auto	Define el orden de solapamiento de acuerdo con sus elementos adyacentes (por defecto)
número	Define el orden que va a tener el orden a la hora de solaparlo con otros elementos. Los valores negativos son permitidos
initial	Pone la propiedad a su valor por defecto
inherit	Hereda este valor del elemento padre

Ejemplo:

```
<!DOCTYPE html>

<html>

 <head>

 <title>z-index</title>
```

```
<meta charset="UTF-8">

<style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 .comunes

 { position: absolute;

 left: 0px;

 top: 0px;

 }

 .debajo

 {

 color:blue;

 width:250px;

 height:200px;

 z-index: -1;

 border: 2px green solid;

 }

 .enmedio

 {

 color:black;

 width:150px;

 height:100px;

 z-index: 0;

 border: 2px black dotted;

 }
```



```
.encima
{
 z-index: 0;

 color: aquamarina;

 border: 2px green dashed;

}

</style>

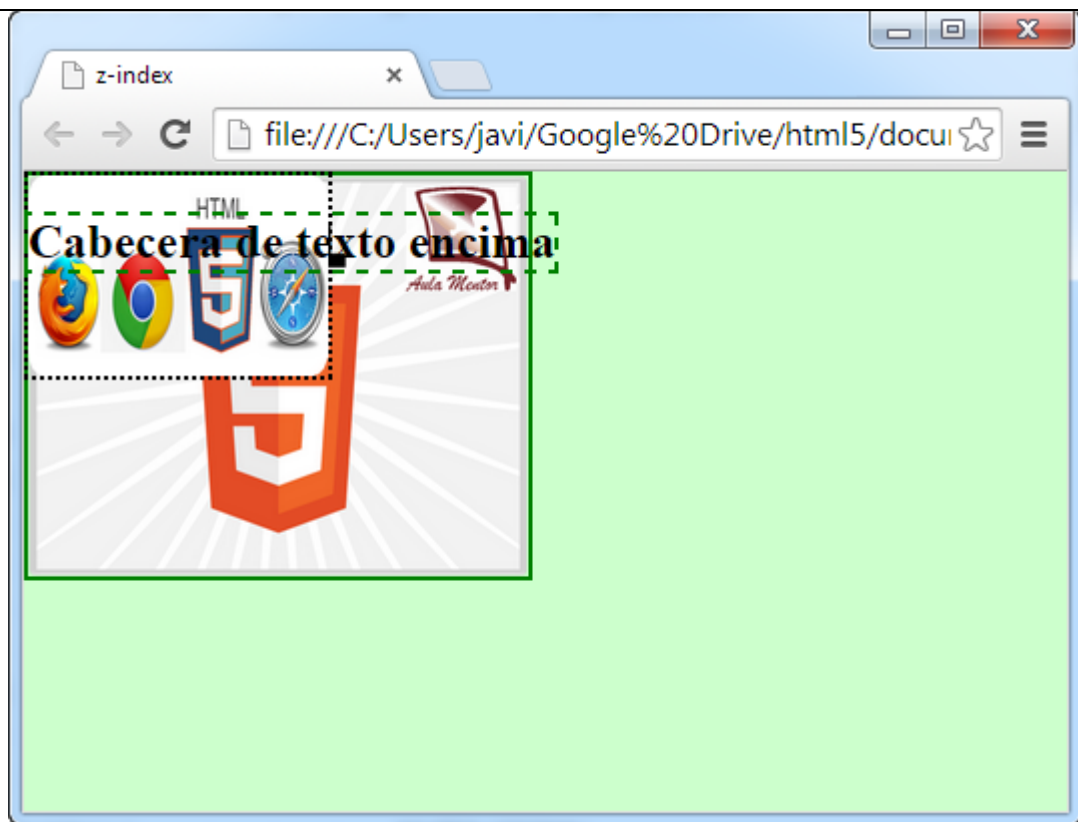
</head>

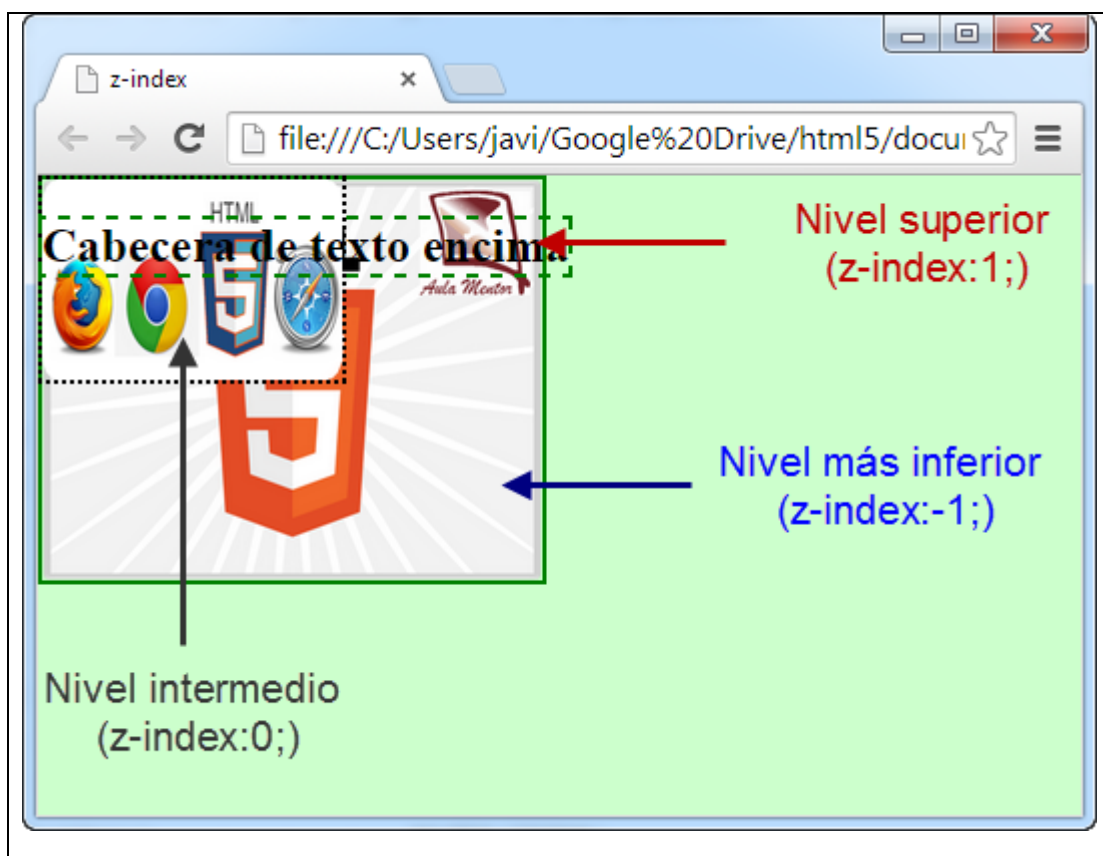
<body class="fondoverde">

<h2 class="encima comunes"> Cabecera de texto encima</h2>

</body>

</html>
```





En este ejemplo la clase “comunes” permite a todos los elementos posicionarse absolutamente en la esquina superior izaquierda de <body> o de la página en general con lo cual tenemos a los tres elementos de la página (dos imágenes y un texto) solapados. Tal y como está escrito el código html, la imagen mentorhtml5.png es la de nivel más inferior al aplicársele en la clase “debajo” el mínimo valor de z-index, la cabecera <h2> está por encima de todas al aplicarle la clase “encima” y tener el mayor valor de z-index de todos los elementos y, por último, la imagen compatibilidad.png está en medio de las anteriores debido a que la clase que se le aplica tiene un valor intermedio de la propiedad z-index entre el de la clase “encima” y el de la clase “debajo”. Comentar también que.

#### 4.9.6. Posicionamiento flotante

Permite que los elementos que se encuentran junto a otro al que se aplica este tipo de posicionamiento, fluyan alrededor de él.

La propiedad **float** permite a un elemento tener un posicionamiento flotante y la propiedad **clear** aplicada a un elemento impide que fluya por el lado o lados especificados alrededor de un elemento flotante

Propiedad	Descripción	Valores
-----------	-------------	---------

float	Indica si un elemento será flotante o no	left, right, both, none (por defecto), initial, inherit
clear	Especifica en qué lado de un elemento los elementos flotantes no serán permitidos	left, right, both, none (por defecto), initial, inherit

```

!DOCTYPE html>

<html>

 <head>

 <title>Posicionamiento flotante</title>

 <meta charset="UTF-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;

 }

 .flotante

 { float: left;

 width: 110px;

 height: 90px;

 margin-left: 50px;

 }

 </style>

 </head>

 <body class="fondoverde">

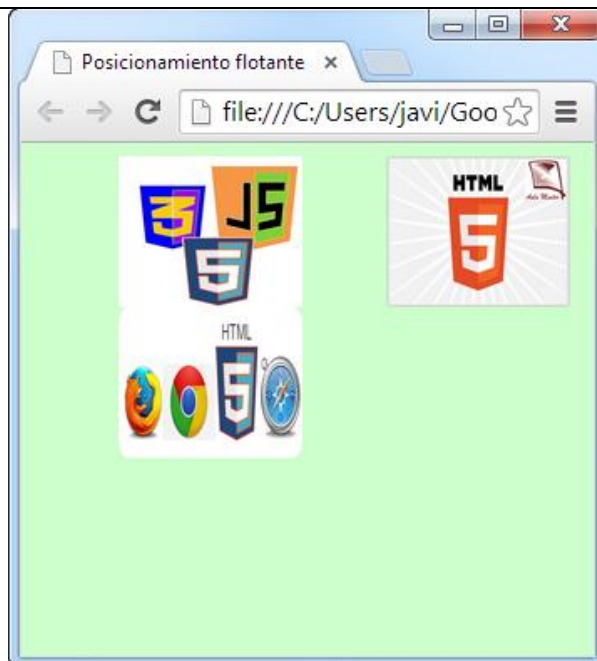
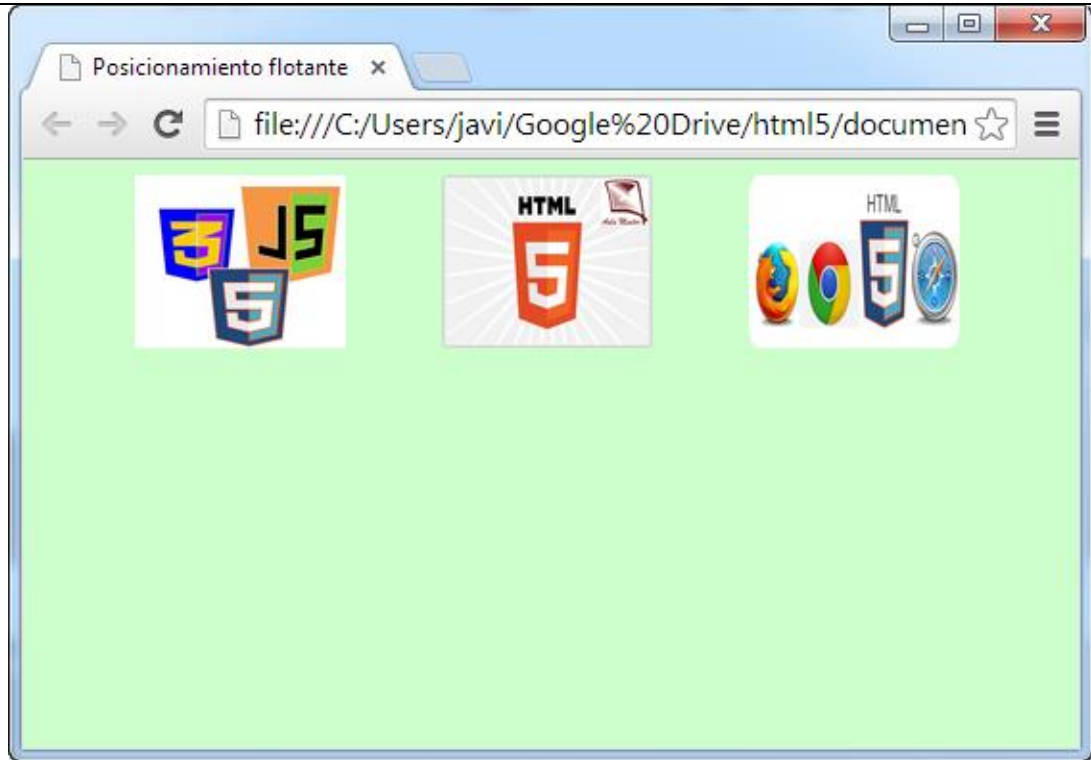

```

```

```

```
</body>
```

```
</html>
```



En este ejemplo asignamos una dimensión y una posición flotante a la izquierda a tres imágenes. En la primera figura vemos como como representaría la página el navegador si en su ancho caben las tres imágenes más el espacio entre las mismas. La segunda figura representa la página cuando se ha hecho más pequeño el ancho del navegador y no caben las tres figuras llevando la que está más a la derecha debajo de la primera imagen de la izquierda (float:left) y, si seguimos reduciendo la anchura del navegador, vemos que la segunda imagen se posiciona a la izquierda debajo de las otras dos.

Otro ejemplo de uso que posee el posicionamiento flotante es para hacer fluctuar texto alrededor de una imagen. En el siguiente ejemplo hacemos que la imagen queda a la izquierda del texto de un párrafo.

```
<html>

 <head>

 <title>Posicionamiento flotante</title>

 <meta charset="UTF-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;

 }

 .flotante

 {float: left;

 width: 110px;

 height: 90px;

 margin: 0px 15px 10px 0px;

 }

 p#párrafo

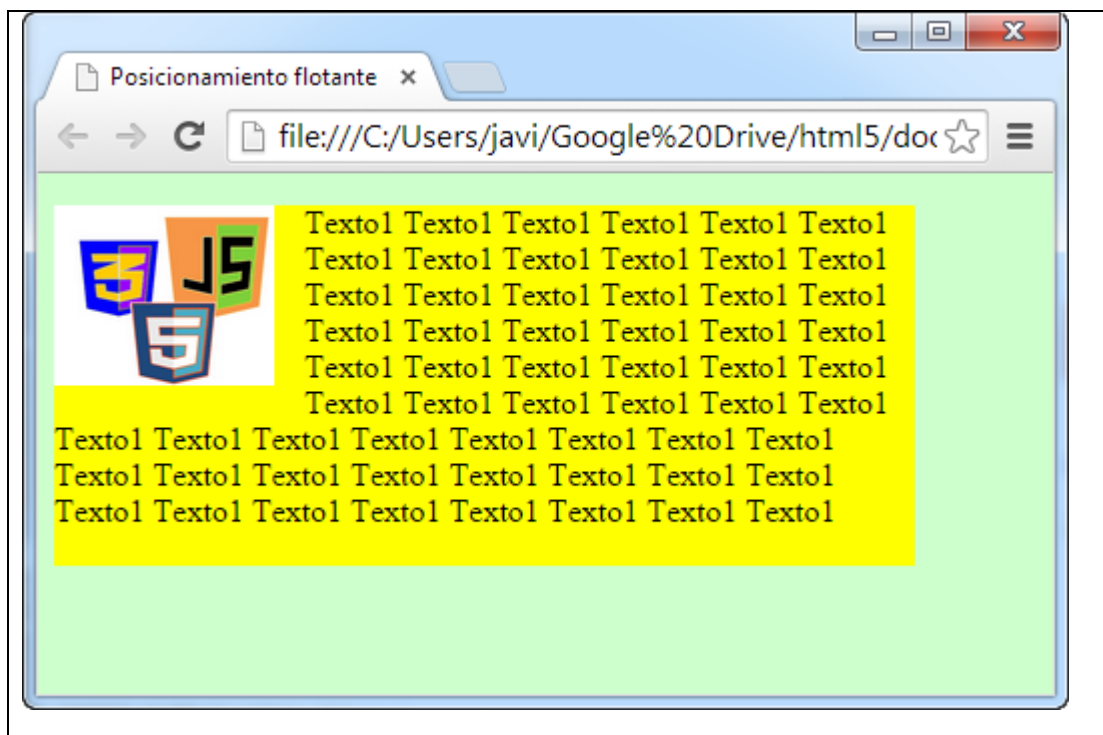
 {
```

```
 width:430px;
 height:180px;
 background-color:yellow;
 }

</style>
</head>
<body class="fondoverde">

<p id="párrafo">

Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
Texto1 Texto1 Texto1 Texto1 Texto1 Texto1
</p>
</body>
</html>
```

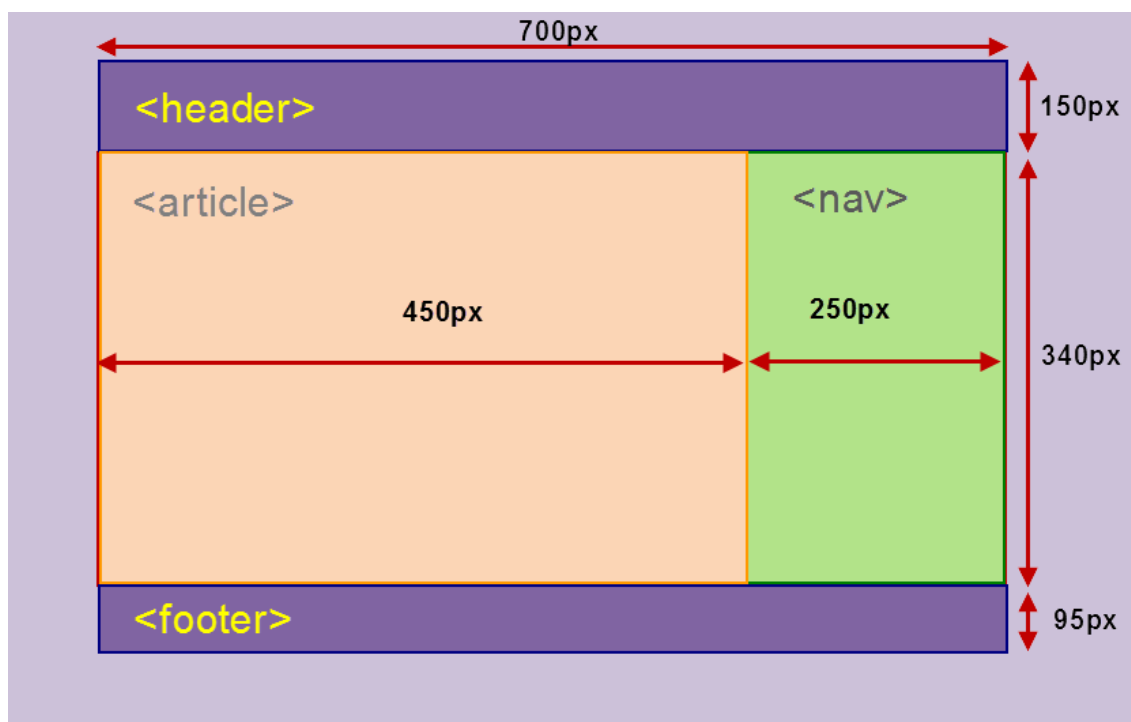


## Ejercicios

Para esta tanda de ejercicios vas a crear el archivo posicionamiento.css donde vas a guardar los estilos que necesites y déjalo en la carpeta **estilos** de tu estructura. Las páginas html síguelas almacenando en la carpeta **tema4**

### Ejercicio 47:

Tomando como base el ejercicio 45 en el que creaste una página web con la siguiente estructura:



Haz lo siguiente:

- En `<header>`
  - Pon el siguiente texto centrado tanto vertical como horizontalmente con letra muy grande y respetando el formato que te muestra a continuación:

**POSICIONAMIENTO DE ELEMENTOS**

- En `<article>`
  - Crea un párrafo, elemento `<p>`, de 100px de largo por 80px de ancho, ponle un color de fondo y escribe bastante texto dentro de él.
  - Crea otro párrafo, elemento `<p>`, de 150px de largo por 90 px de anchura, ponle otro color de fondo y escribe bastante texto dentro de él

Observa cómo queda y guarda el ejercicio como ejercicio47.html.

#### Ejercicio 48:

Modifica el ejercicio anterior y pon el primer párrafo a 15 pixels del borde izquierdo y a 20 pixels del borde superior del elemento `<article>` ¿Cómo queda el segundo párrafo?

#### Ejercicio 49:

Utilizando el ejercicio anterior crea dos clases que, aplicándolas al segundo párrafo, lo pongan a 60 pixels por debajo del párrafo anterior una de ellas y 60



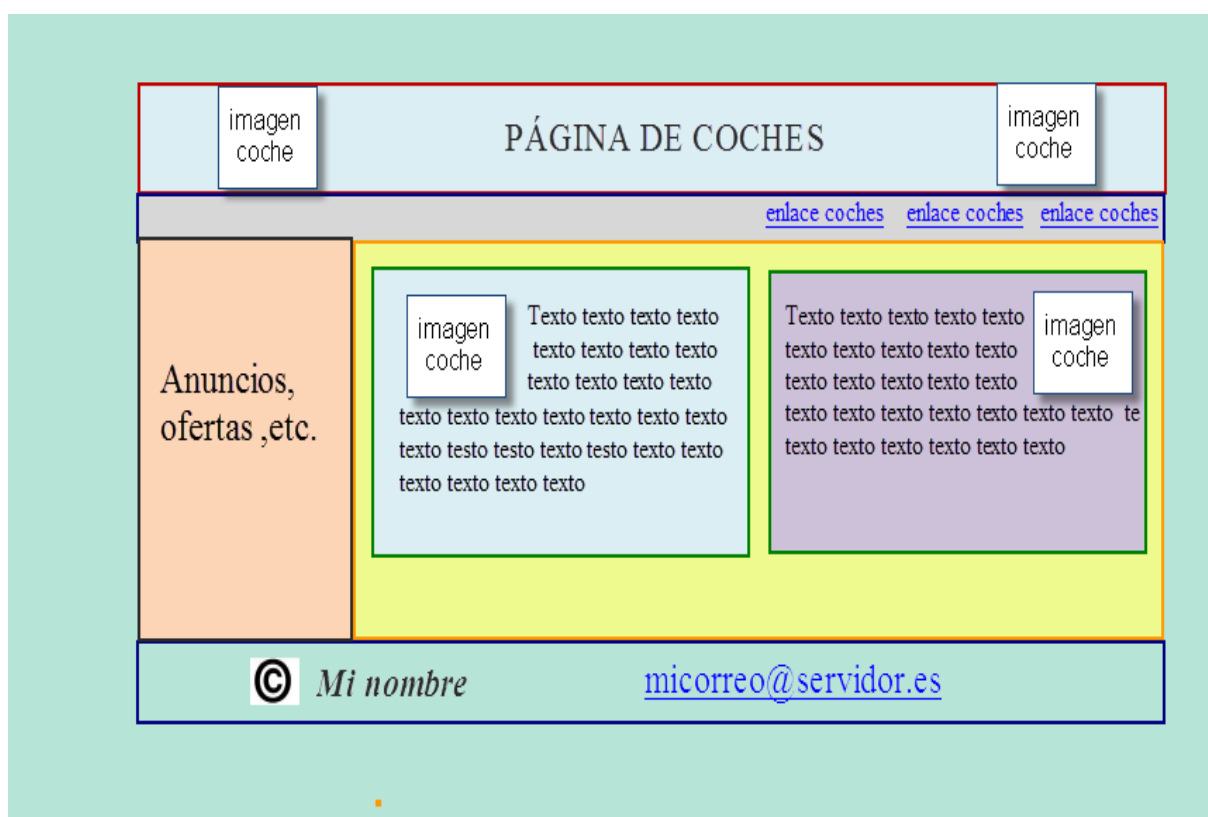
pixels del borde superior y 40 pixels del borde derecho de <article> la otra ¿Qué tipos de posicionamiento has usado en cada caso?

### Ejercicio 50:

Haz que el texto de los dos párrafos se coloque a 5 pixels del borde superior, a 3 del borde izquierdo a 6 del borde derecho y a 2 del borde inferior del párrafo (procura usar en lo posible la sintaxis alternativa para los valores de los estilos). Guarda el fichero modificado como ejercicio50.html.

### Ejercicio 51:

Vamos a hacer una página simple de coches con este formato y siguiendo las siguientes pautas



- Pon los colores de fondo que creas convenientes a cada uno de los elementos incluido <body>.
- Para los textos emplea el tipo de letra y el tamaño que quieras.
- La imagen anterior es un elemento <section> centrado y que ocupa el 85% del ancho de la página.
- Dentro de <section>
  - <header>
    - 150 pixels de altura.

- Una imagen de coches a 10 pixels del borde izquierdo que ocupe 70 pixels de ancho y 150 de alto.
- Un texto centrado en letra arial, negrita y con caracteres de 15 puntos.
- Otra imagen diferente a la anterior a 10 pixels del borde derecho que ocupe 70 pixels de ancho y 150 de alto.
- <nav>
  - 60 pixels de altura.
  - Tres enlaces a páginas de coches a 5 pixels del borde derecho y centrados verticalmente.
- <aside>
  - 30% de ancho.
  - 400 pixels de altura
  - Pon anuncios de coches e información (si quieres puedes emplear algún elemento como <p> para contener el texto)
- <section>
  - 70% de ancho.
  - 400 pixels de altura
  - <article>
    - 200 pixels de ancho
    - 350 pixels de altura
    - Situado a 5 pixels del borde izquierdo y a 15 pixels del borde superior de su contenedor <section>
    - Una imagen de 40x40 de coches que esté a 2 pixels de borde izquierdo y a 3 del superior de <article>
    - Texto que fluya alrededor de la imagen que esté a 3 pixels de borde superior a 2 del borde izquierdo, a 4 del borde derecho y a 7 del borde inferior
  - <article>
    - 200 pixels de ancho
    - 350 pixels de altura
    - Situado a 5 pixels del borde derecho y a 15 pixels del borde superior de su contenedor <section>
    - Una imagen de 40x40 de coches que esté a 2 pixels de borde derecho y a 3 del superior de <article>
    - Texto que fluya alrededor de la imagen que esté a 3 pixels de borde superior a 2 del borde izquierdo, a 4 del borde derecho y a 7 del borde inferior
- <footer>
  - 100 pixels de altura
  - Texto centrado verticalmente a 20 pixels del borde izquierdo (puedes usar algún elemento para poner el texto)

- Enlace a tu correo a 70 pixels del borde derecho y centrado verticalmente.
- Guarda el archivo como ejercicio51.html y agrega a posicionamiento.css los estilos o modifica los que hubiera.

## 4.10. LISTAS

En HTML5 existen tres tipos de listas: las desornadas (<ul>), la ordenadas (<ol>) y las de definición (<dl>) que se diferencian en la forma que tienen marcar los elementos de cada una de ellas. A través de CSS podemos ampliar el número de símbolos marcadores tanto de las listas ordenadas como desordenadas e, incluso, poner imágenes en vez de esos símbolos.

### 4.10.1. Propiedades de listas

Indica el símbolo o imagen marcador del elemento de una lista ordenada o desordenada así como su posición con respecto al propio elemento o al contenedor

Propiedad	Descripción	Valores
list-style	Sintaxis alternativa o shorthand	<i>list-style-type list style-position list-style-image</i>
list-style-type	Define la forma del símbolo marcador	none, circle, square, upper-roman, lower-alpha, decimal, ...
list-style-image	Especifica una imagen como símbolo marcador	none, url ( <i>imagen</i> ), initial, inherit
list-style-position	Indica la posición del símbolo marcador	inside, outside, initial, inherit

**Ejemplo de *list-style-type*:**

```

<!DOCTYPE html>
<html lang="es-es">
<head>
 <meta charset="UTF-8">
 <title>Listas CSS</title>

 <style type="text/css">
 .fondoverde {background-color:#ccffcc;}

 ul.círculo { list-style-type: circle; }

 ul.cuadrado { list-style-type: square; }

 ol.mayúsculasromanas { list-style-type: upper-roman; }

 ol.minúsculasalfabéticas { list-style-type: lower-alpha; }

 </style>
</head>
<body class="fondoverde">
 <h3> Lista desordenada con círculos</h3>
 <ul class="círculo">
 Gato
 Perro
 Caballo

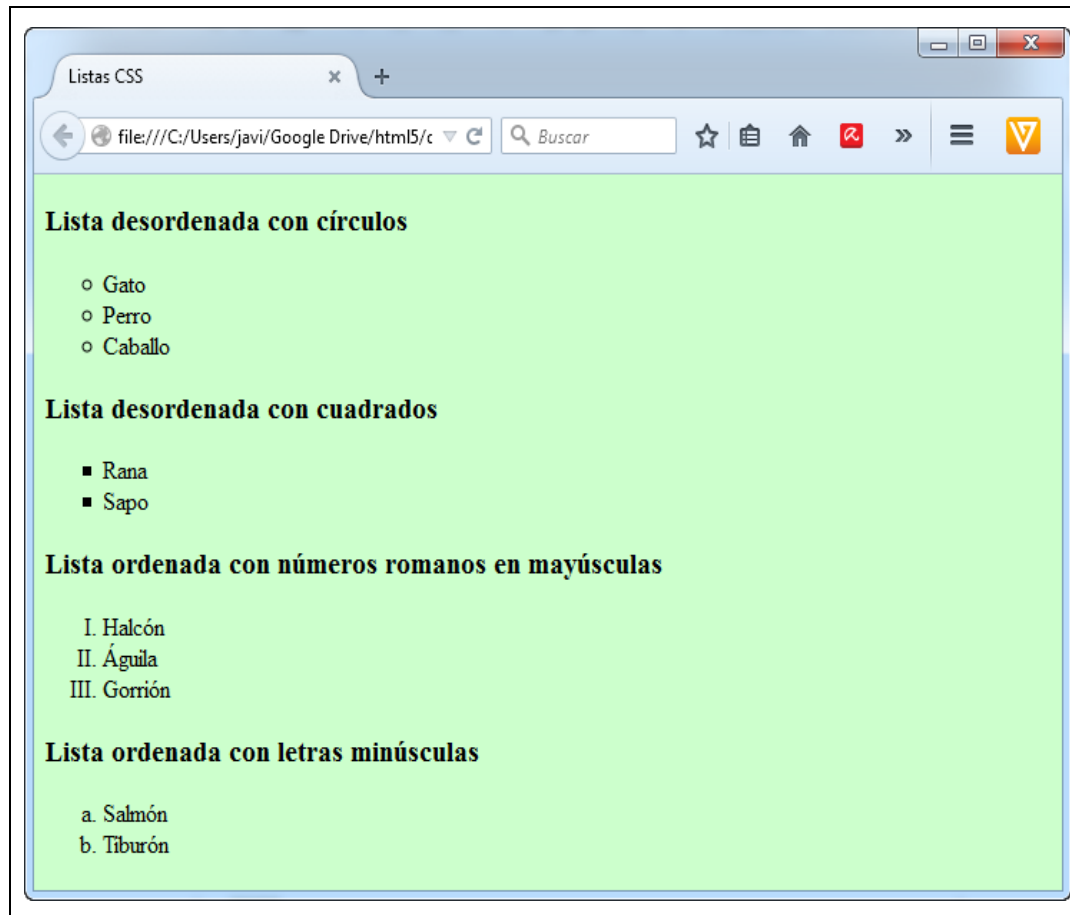
 <h3> Lista desordenada con cuadrados</h3>
 <ul class="cuadrado">
 Rana
 Sapo

 <h3> Lista ordenada con números romanos en mayúsculas </h3>
 <ol class="mayúsculasromanas">
 Halcón
 Águila
 Gorrión

 <h3> Lista ordenada con letras minúsculas </h3>
 <ol class="minúsculasalfabéticas">
 Salmón
 Tiburón

</body>
</html>

```



### Ejemplo de list-style-image:

```
<!DOCTYPE html>
<html lang="es-es">
<head>
 <meta charset="UTF-8">
 <title>Listas CSS con imágenes</title>

 <style type="text/css">
 .fondoverde {background-color:#ccffcc;}

 ul {padding-left:150px;}

 ul.caballo { list-style-image: url("iconocaballo.jpg");}

 ul.torre { list-style-image: url("iconotorre.jpg"); }
```

```
</style>
</head>
<body class="fondoverde">
 <h3> Lista desordenada con una la imagen de un caballo</h3>
 <ul class="caballo">
 Gato
 Perro
 Caballo

 <h3> lista desordenada con la imagen de torre</h3>
 <ul class="torre">
 Rana
 Sapo

</body>
</html>
```



En el ejemplo anterior hay que reseñar que iconocaballo.jpg e iconotorre.jpg tienen dimensiones diferentes por eso se ven de manera distinta.

	<i>No se puede poner cualquier imagen como símbolo de una lista ya que se representa con su tamaño original por lo que hay que usar imágenes de tipo icono de dimensiones muy pequeñas.</i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



Existe otra manera de poner imágenes pequeñas o iconos como símbolos de listas y es a través del fondo o background de los elementos <li>

**Ejemplo:**

```
<!DOCTYPE html>
<html lang="es-es">
<head>
 <meta charset="UTF-8">
 <title>Listas CSS con imágenes</title>

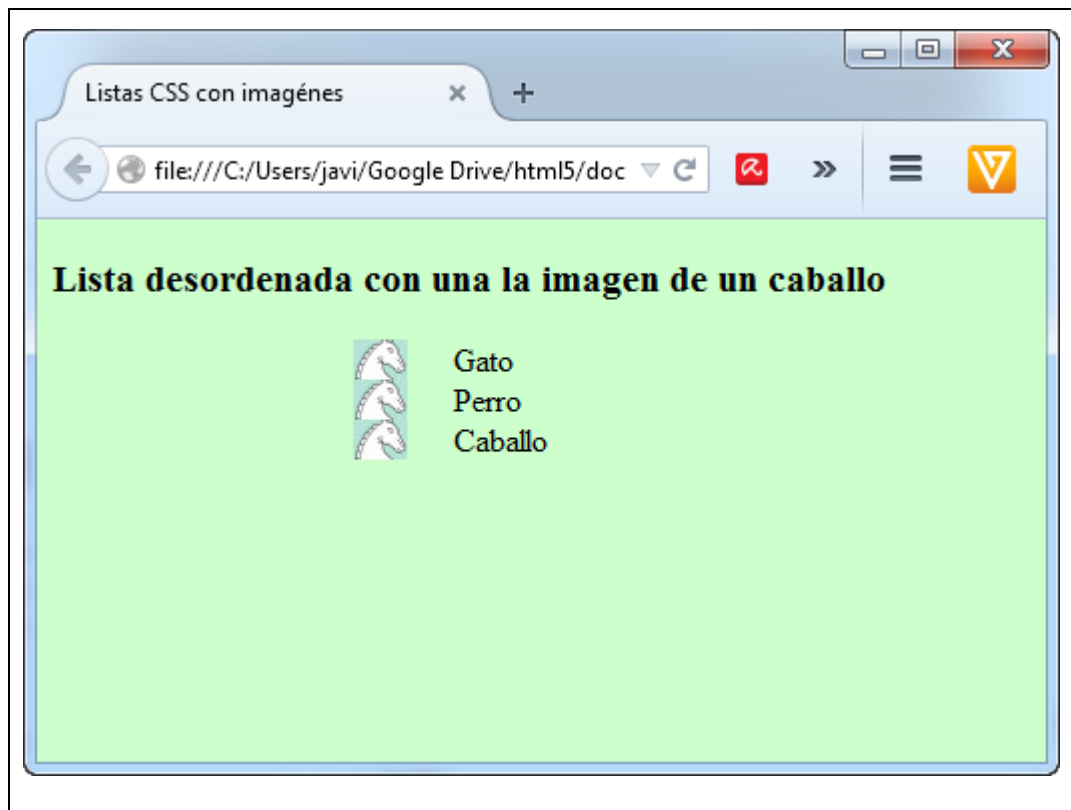
 <style type="text/css">
 .fondoverde {background-color:#ccffcc;}
 ul {padding-left:150px;
 list-style-type:none;}

 ul li { background-image: url("iconocaballo.jpg");
 background-repeat: no-repeat;
 background-position: 0px center;
 padding-left: 50px; }

 </style>
</head>
<body class="fondoverde">
 <h3> Lista desordenada con una la imagen de un caballo</h3>

 Gato
 Perro
 Caballo

</body>
</html>
```



Fijándonos en los estilos vemos por un lado que aplicamos a todos los elementos `<ul>` la propiedad **`list-style-type:none`** lo que implica que no se usa ningún símbolo o imagen para los elementos de la lista. En cambio, para los elementos `<li>` que pertenezcan a cualquier `<ul>` asignamos una imagen como background (`iconocaballo.jpg`) y jugamos con las propiedades de background y de espacio (`padding`) para colocarlo en el lugar adecuado y separar la imagen del texto de los elementos de lista.



*Al igual que con **`list-style-image`**, usa imágenes de tipo icono para los símbolos de lista*

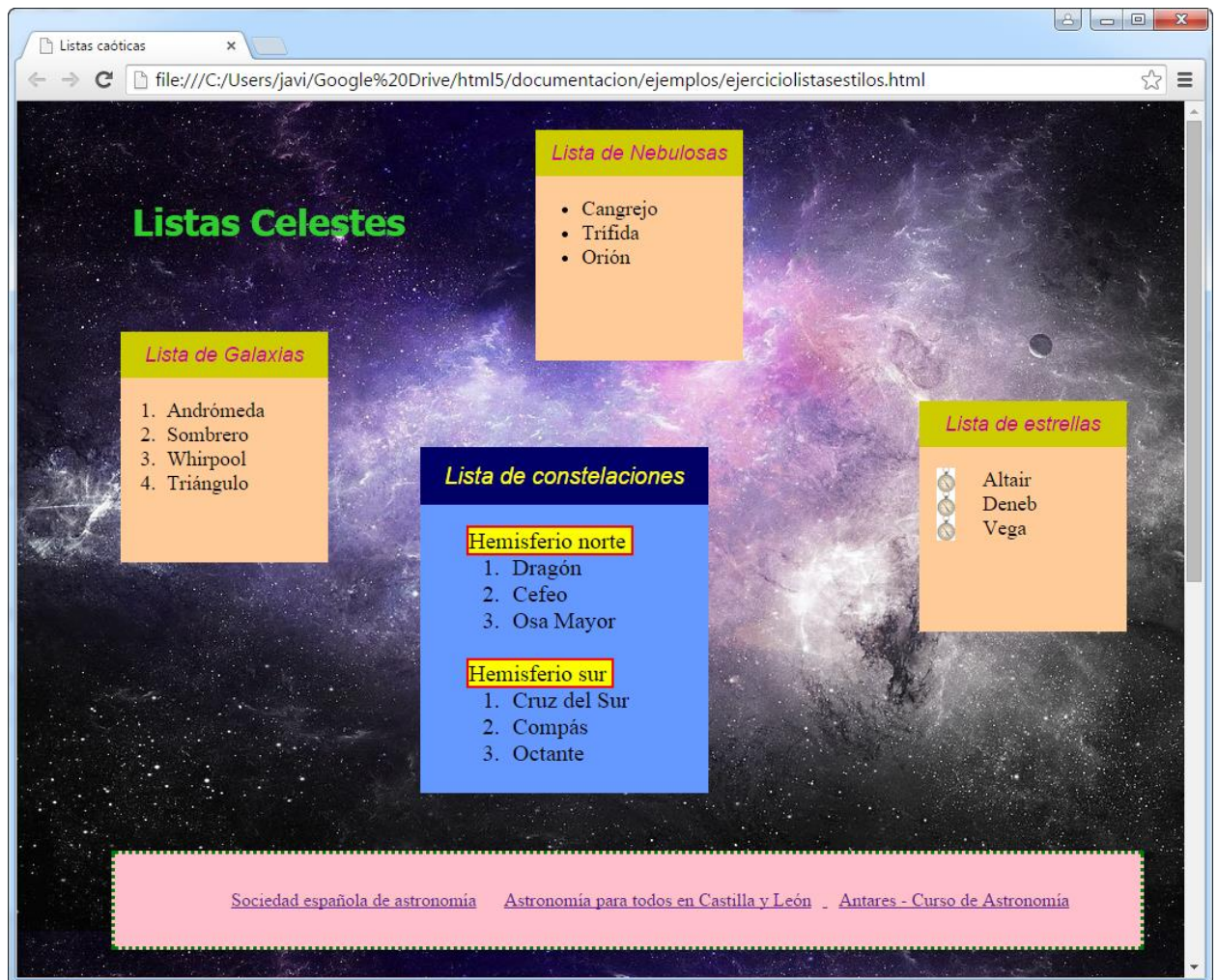
**Ejercicios:**

Para esta tanda de ejercicios vas a crear el archivo listas.css donde vas a guardar los estilos que necesites y déjalo en la carpeta **estilos** de tu estructura. Las páginas html síguelas almacenando en la carpeta **tema4**.

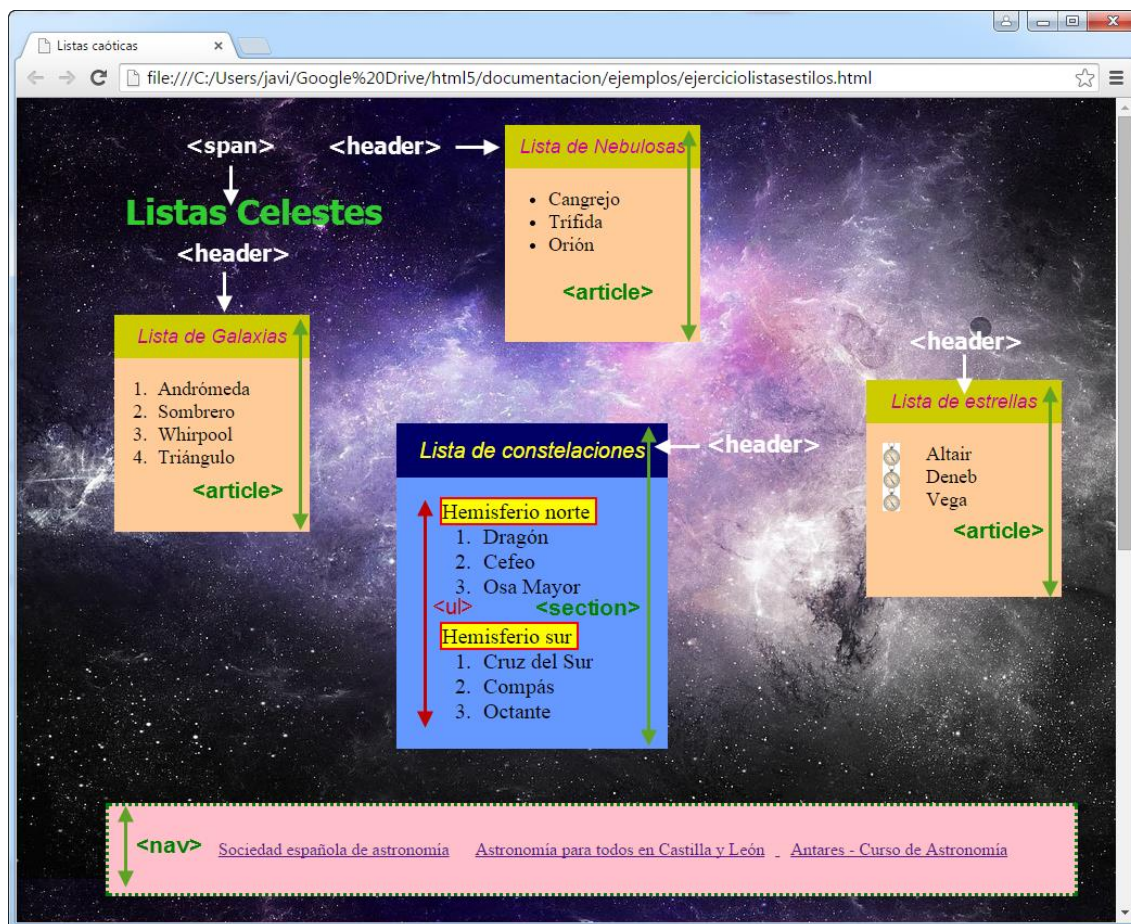
También, a partir de estos ejercicios, vamos a ver la forma a afrontar la realización de páginas web con estilos. Para ello vamos a seguir los siguientes pasos:

- Hacer un dibujo aproximado del aspecto que queremos conseguir en la página html con su contenido o con parte de él.
- Crear la página web solamente con el contenido, es decir, con todos los elementos que vamos a emplear y la información que va a tener el documento html sin aplicarles ningún estilo.
- Crear los estilos y aplicarlos a los elementos correspondientes.

**Ejercicio 52:** Vamos a crear la siguiente página con listas colocadas de forma caótica



Un posible dibujo indicando los elementos de la página podría ser este:



Y, a partir de la imagen anterior, creamos el archivo ejercicio52.html que es el contenido de la página con los elementos que vamos a utilizar sin aplicar estilos.

```
<!DOCTYPE html>
<html lang="es-es">
<head>
 <meta charset="UTF-8">
 <title>Listas caóticas</title>
</head>
<body>
 Listas Celestes
 <article>
 <header> Lista de estrellas </header>

```

```
Altair
Deneb
Vega

</article>
<article>
 <header> Lista de Galaxias </header>

 Andrómeda
 Sombrero
 Whirpool
 Triángulo

</article>
<article>
 <header> Lista de Nebulosas </header>

 Cangrejo
 Trífida
 Orión

</article>
<section>
 <header> Lista de constelaciones</header>

 Hemisferio norte

 Dragón
 Cefeo
 Osa Mayor

 Hemisferio sur

```



```

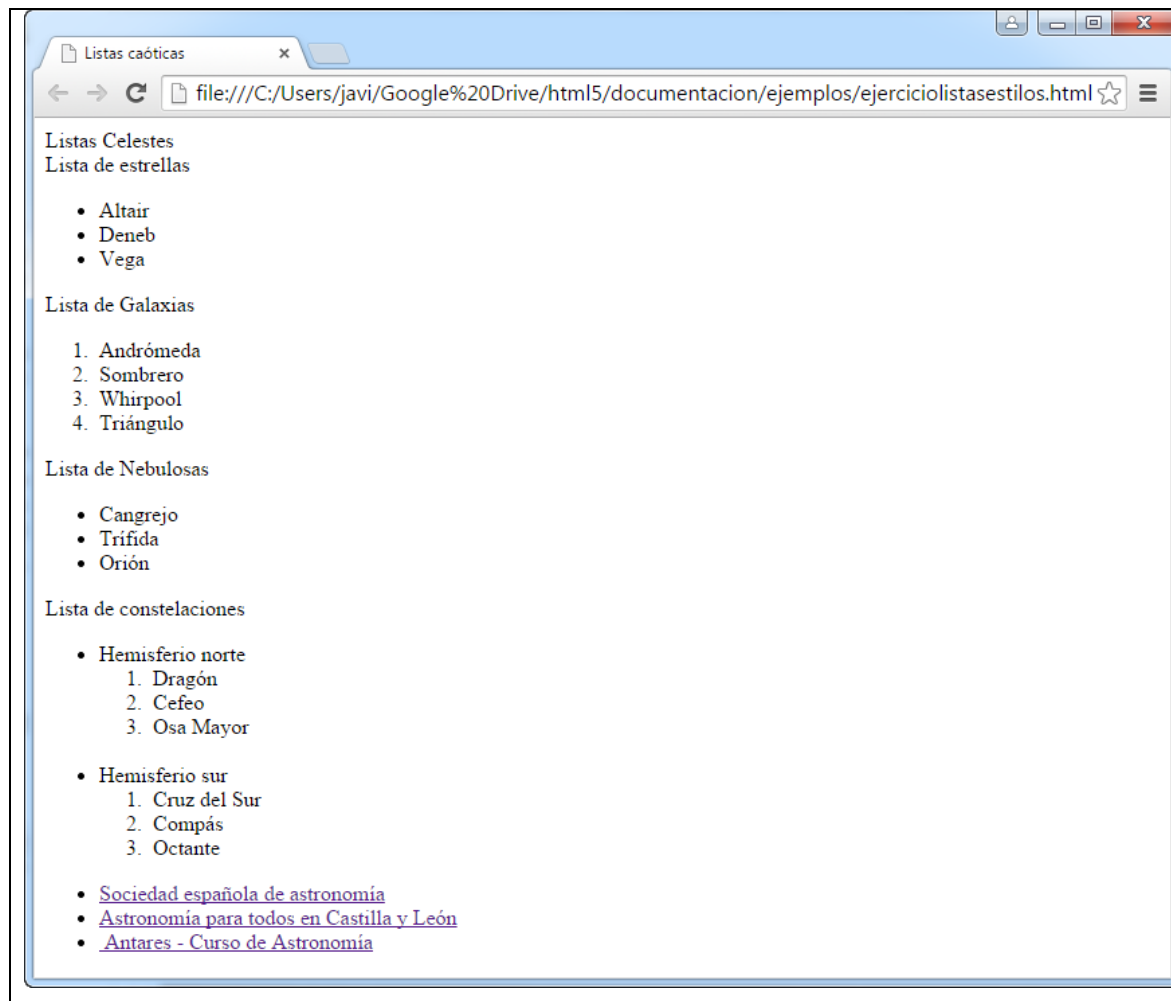
 Cruz del Sur
 Compás
 Octante

</section>
<nav>

 Sociedad española de
astronomía

<a
href="http://www.educa.jcyl.es/educacyl/cm/gallery/Recursos%20Infinity/aplic
aciones/astronomia/index.html"> Astronomía para todos en Castilla y León
 Antares - Curso de
Astronomía

</nav>
</body>
</html>
```



En la imagen anterior vemos como queda la página que hemos creado en un navegador sin aplicar ningún estilo por lo que vas a crear el archivo de estilos listas.css que agregarás al fichero ejercicio52.html para que te quede como la primera imagen. Ten en cuenta lo siguiente:

- El posicionamiento de todos los elementos es absoluto.
- Pon un fondo “estelar” a la página.
- “Listas Celestes” es un texto con las siguientes características:
  - Fuente: Tahoma.
  - Tamaño de letra: 32 pixels.
  - Color: #33CC33
  - Negrita
  - Posición absoluta: 100 pixels izquierda y 85 pixels desde el borde superior.



- General para todos los artículos, es decir, para los contenedores de las listas de estrellas, galaxias y nebulosas
  - Fuente: Time New Roman
  - Tamaño de letra: 18 pixels.
  - Anchura: 180 pixels
  - Altura: 200 pixels
  - Posicionamiento absoluto.
  - Color de fondo: #FFCC99
- General para todas las cabeceras (<header>) de los artículos
  - Fuente: Arial
  - Estilo de la fuente: Itálica.
  - Alineación central tanto vertical como horizontal
  - Color de fondo: #CCCC00
  - Altura: 40 pixels
- Artículo con la lista de galaxias
  - Posición: 90 pixels a la izquierda y 200 pixels desde el borde superior.
- Artículo con la lista de nebulosas
  - Posición: 450 pixels a la izquierda y 25 pixels desde el borde superior.
- Artículo con la lista de estrellas
  - Posición: 50 pixels desde la derecha y 260 pixels desde el borde superior.
- Lista de estrellas
  - El símbolo de cada elemento de la lista es una imagen.
- Sección de constelaciones
  - Fuente: Times
  - Tamaño de letra: 20 pixels
  - Anchura: 250 pixels
  - Altura: 300 pixels
  - Color de fondo: #6699FF
  - Posición: 350 pixels desde el borde izquierdo y 300 pixels desde el borde superior.
- Elemento <header> de la sección de constelaciones

- Fuente: Arial
- Estilo de fuente: Itálica
- Alineación del texto: Centrada tanto vertical como horizontalmente.
- Color de letra: Amarillo
- Color de fondo: #000066
- Altura: 50 pixels.
- Los elementos `<span>` de la lista de constelaciones
  - Borde sólido de 2 pixels de color rojo.
  - Color de fondo: Amarillo
  - Elemento `<nav>`
  - Altura: 80 pixels
  - Anchura: 850 pixels
  - Posición: 650 pixels borde superior y 50 pixels de borde izquierdo
- Elemento `<ul>` de `<nav>`
  - Borde de puntos de 3 pixels verde
  - Color de fondo: rosa
  - Altura: 80 pixels.
  - Alineación del texto: centrado tanto vertical como horizontalmente.
- Elementos `<li>` de `<ul>` de `<nav>`
  - Espaciado izquierdo y derecho: 10 pixels
  - Visualización en línea.

### Ejercicio 53:

Cambia el tamaño de la ventana del navegador y examina como cambian la disposición de los elementos para que veas la rigidez que tiene el posicionamiento absoluto.

## 4.11. ALGUNAS INNOVACIONES CSS3

CSS3 incorpora nuevas propiedades que mejoran el aspecto de los elementos HTML.

#### 4.11.1. CSS3 Borders

Con CSS3 se pueden crear bordes redondeados, añadir sombras a las cajas y usar una imagen como borde.

Propiedad					
<b>border-radius</b>	9.0	5.0 4.0 -webkit-	4.0 3.0 -moz-	5.0 3.1 -webkit-	10.5
<b>box-shadow</b>	9.0	10.0 4.0 -webkit-	4.0 3.5 -moz-	5.1 3.1 -webkit-	10.5
<b>border-image</b>	11.0	16.0 4.0 -webkit-	15.0 3.5 -moz-	6.0 3.1 -webkit-	15.0 11.0 -o-

Las palabras **-webkit-**, **-moz-** y **-o-** son prefijos que se ponen delante de la propiedad y sirven para indicar la versión del navegador a partir de la cual se reconocen estas propiedades. Por ejemplo:

```
header {
 -webkit-border-radius:30px;
 -moz-border-radius:25px;
 -o-border-radius: 20px;
 border-radius:15 px; }
```

Aplicaría un borde redondeado a los elementos <header> del documento con los siguientes grosores:

- 30 pixels para los navegadores Chrome superiores a la versión 4.0 y Safari 3.1
- 25 pixels para navegadores Mozilla Firefox superiores a la versión 3.5.
- 20 pixels para los navegadores Ópera 15.0 o superiores
  - 15 pixels para todas las versiones de IExplorer a partir de la 9.0, para las versiones de Chrome a partir de la 5.0., para las versiones superiores a la 4.0 de Firefox, la 5.0 de Safari o superiores y la 10.5 o superior de Ópera.

- **border-radius**

Propiedad por la cual se redondea los bordes de una caja.

Este redondeo se hace especificando una cantidad, ya sea en unidades o en %, que indica la curvatura de las esquinas.

En realidad, **border-radius** es sintaxis alternativa o “shorthand” que permite el redondeo de las cuatro bordes de una caja por lo que `border-radius: 10px;` equivale a:

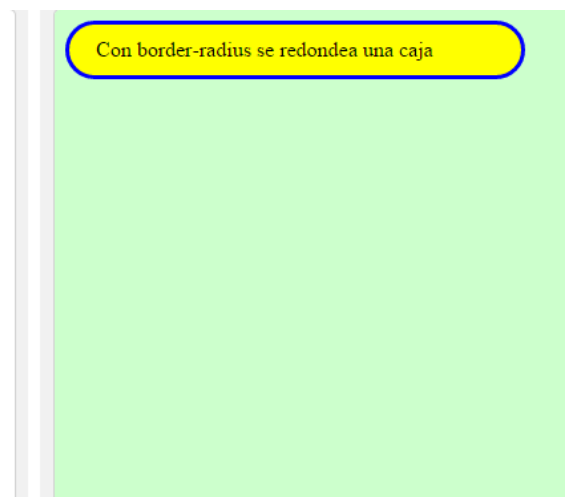
```
border-top-left-radius: 10px;
border-top-right-radius: 10px;
border-bottom-right-radius: 10px;
border-bottom-left-radius: 10px;
```

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<style>
body { background-color: #ccffcc; }
article {
 border: 3px solid blue;
 padding: 10px 20px;
 background: yellow;
 width: 300px;
 border-radius: 25px;
}
</style>
</head>
<body>

<article> Con border-radius se redondea una caja
</article>

</body>
</html>
```



- **box-shadow**

Añade sombra a las cajas indicando la cantidad y color. Los parámetros requeridos son el desplazamiento vertical, el desplazamiento horizontal y el color de dichas sombra

```

<!DOCTYPE html>
<html>
<head>
<style>
body { background-color:#ccffcc;}
article {
border: 3px solid blue;
padding: 10px 20px;
background: yellow;
width: 320px;
border-radius: 25px;
box-shadow: 40px 10px #888888;
}
</style>
</head>
<body>

<article> Con box-shadow se añade sombra a una caja
</article>

</body>
</html>

```

Con box-shadow se añade sombra a una caja

En este caso la sombra está desplazada 40 pixels a la izquierda (desplazamiento horizontal) y 10 pixels hacia abajo (desplazamiento vertical) con un color gris.

- **border-image**

Asigna una imagen como borde de una caja.

```

<!DOCTYPE html>
<html>
<head>
<style>
body {background-color:#ccffcc;}
section {
border: 20px solid transparent;
width: 270px;
padding: 10px 20px;
}

#repetir { border-image: url(border.png) 30 30 round; }

#estirar {border-image: url(border.png) 30 30 stretch;}
</style>
</head>
<body>

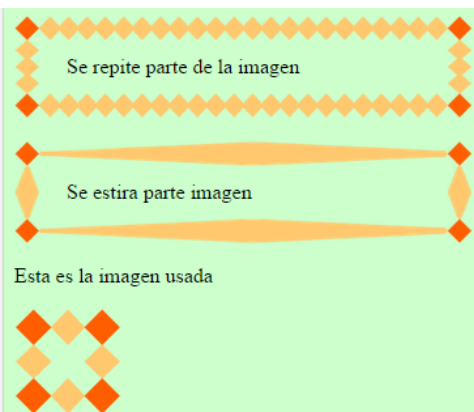
<section id="repetir">
Se repite parte de la imagen
</section>

<section id="estirar">Se estira parte imagen</section>

<p>Esta es la imagen usada</p>

</body>
</html>

```



#### 4.11.2. Backgrounds

CSS3 aporta nuevas propiedades para fondos de elementos como son control de tamaños, posición y múltiples imágenes.

Propiedad					
<b>background-size</b>	9.0	4.0 1.0 -webkit-	4.0 3.6 -moz-	5.0 3.0 -webkit-	10.5 10.0 -o-
<b>background-origin</b>	9.0	1.0	4.0	3.0	10.5

- **background-size**

Permite el cambio de tamaño visual de una imagen de fondo.

**Ejemplo:**

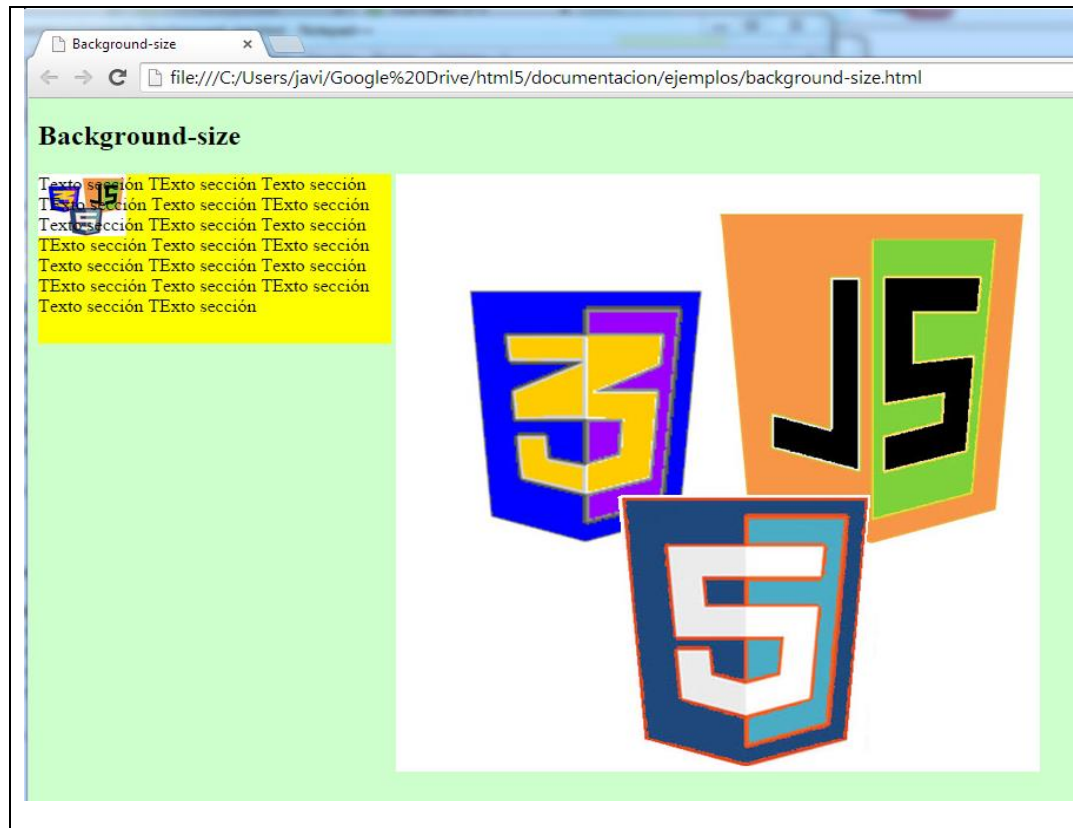
```

<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>Background-size</title>
 <meta charset="UTF-8">
 <style type="text/css">
 body {background-color:#ccffcc;}
 section {
 background: url("composicion.jpg");
 background-size: 75px 55px;
 background-color:yellow;
 background-repeat:no-repeat;
 width:300px;
 height:150px;
 display:inline-block;
 vertical-align: top;
 }
 </style>
</head>
<body>
 <section>
 </section>
</body>
</html>

```

```
 }
 img { display:inline-block;}
 </style>
</head>
<body>
 <h2> Background-size </h2>
 <section>
 Texto sección TExto sección Texto sección TExto sección
 Texto sección TExto sección Texto sección TExto sección
 Texto sección TExto sección Texto sección TExto sección
 Texto sección TExto sección Texto sección TExto sección
 </section>

</body>
</html>
```



En este ejemplo se ve que la imagen “composición.jpg” es la imagen de fondo del elemento <section> y que hemos modificado su tamaño visual. Su tamaño real puede ver al lado.

- **background-origin**

Indica en que parte de la caja se inserta la imagen de fondo. Esta parte es una de la siguientes: border-box, padding-box y content-box.



**border-box**

**padding-box**

**content-box**

**Ejemplo:**

```
<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>Background-origin</title>
 <meta charset="UTF-8">
 <style type="text/css">
 body {background-color:#ccffcc;}

 section {
 border: 10px solid green;
 padding: 50px;
 background: url("composicion.jpg");
 background-size: 45px 35px;
 background-color:yellow;
 background-repeat:no-repeat;
```

```
 background-position: left;

 width:150px;

 height:100px;

 display:inline-block;

 vertical-align: top;

 }

 #uno { background-origin:border-box;}

 #dos { background-origin:padding-box;}

 #tres {background-origin:content-box;}

</style>
</head>
<body>

 <h2> Background-size </h2>

 <section id="uno">

 <h3>border-box</h3>

 Texto sección TTexto sección

 </section>

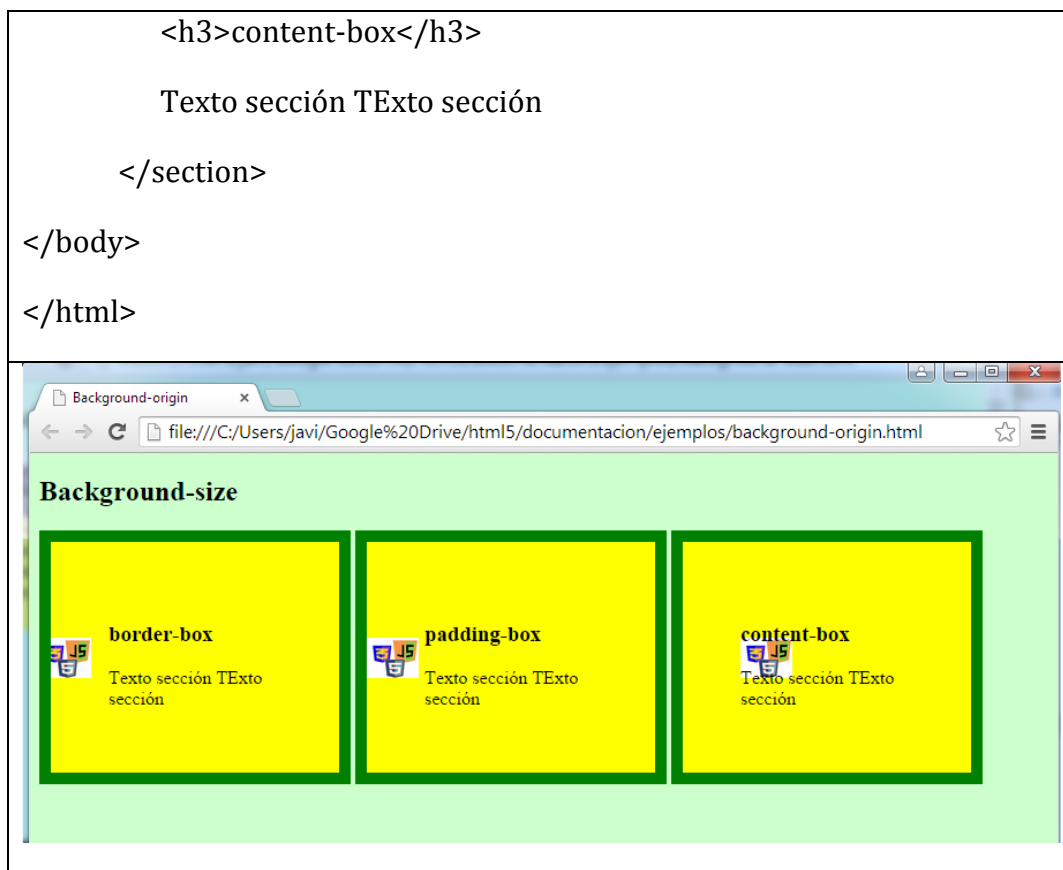
 <section id="dos">

 <h3>padding-box</h3>

 Texto sección TTexto sección

 </section>

 <section id="tres">
```



En este ejemplo, en la imagen de la sección de la izquierda, la imagen de fondo está definida en el borde la sección, en la central está en el espacio entre el contenido y el borde y, por último, en la tercera está en el contenido de la sección.

#### 4.11.3. Gradientes

Los gradientes CSS3 permiten realizar transiciones suaves entre dos o más colores especificados.

Propiedad					
<b>linear-gradient</b>	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.1 -o-
<b>radial-gradient</b>	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.6 -o-

<b>repeating-linear-gradient</b>	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.1 -o-
<b>repeating-radial-gradient</b>	10.0	26.0 10.0 -webkit-	16.0 3.6 -moz-	6.1 5.1 -webkit-	12.1 11.6 -o-

- **linear-gradient**

A partir de dos colores se hacen transiciones suaves entre ellos en bandas horizontales dentro de las cuales se puede establecer un punto de partida, una dirección o un ángulo.



***linear-gradient** es una propiedad de **background** y su sintaxis es:*

**background: linear-gradient(dirección, color1, color2, ....)**

```
<!DOCTYPE html>
<html>
<head>
<style>
 body {background-color:#ccffcc;}
 #grad1 {
 height: 100px;
 background: linear-gradient(yellow, blue);
 }
 #grad2 {
 height: 100px;
 background: linear-gradient(90deg, yellow,
blue);}
</style>
</head>
<body>

<h3>Linear Gradient </h3>
<p>Comienza al principio del elemento y cambia de
amarillo a azul</p>

<div id="grad1"></div>

<p>En este caso se asigna un ángulo de 90 grados
para hacer la transición</p>
<div id="grad2"></div>

</body>
</html>
```

### Linear Gradient

Comienza al principio del elemento y cambia de amarillo a azul



En este caso se asigna un ángulo de 90 grados para hacer la transición



- **radial-gradient**

A través de esta propiedad de **background** se pueden definir transiciones circulares de colores.

```

<!DOCTYPE html>
<html>
<head>
<style>
body {background-color:#ccffcc;}
#grad1 {
height: 100px;
background: radial-gradient(yellow, blue,red);
}

#grad2 {
height: 100px;
background: radial-gradient(yellow 5%, blue 15%
,red 60%);}
</style>
</head>
<body>

<h3>Radial Gradient </h3>
<p>Comienza desde el centro del elemento y se
dibujan círculos concéntricos que ocupan lo mismo
</p>

<div id="grad1"></div>

<p>En este caso el color rojo ocupa el 60%, el azul
el 15% y el amarillo el 5%</p>

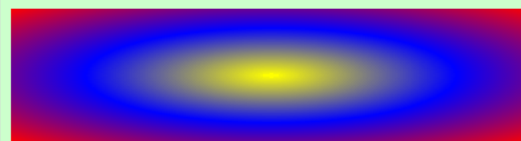
<div id="grad2"></div>

</body>
</html>

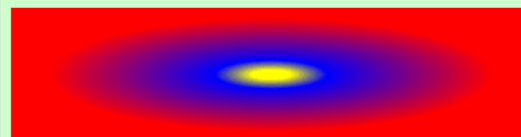
```

### Radial Gradient

Comienza desde el centro del elemento y se dibujan círculos concéntricos que ocupan lo mismo



En este caso el color rojo ocupa el 60%, el azul el 15% y el amarillo el 5%

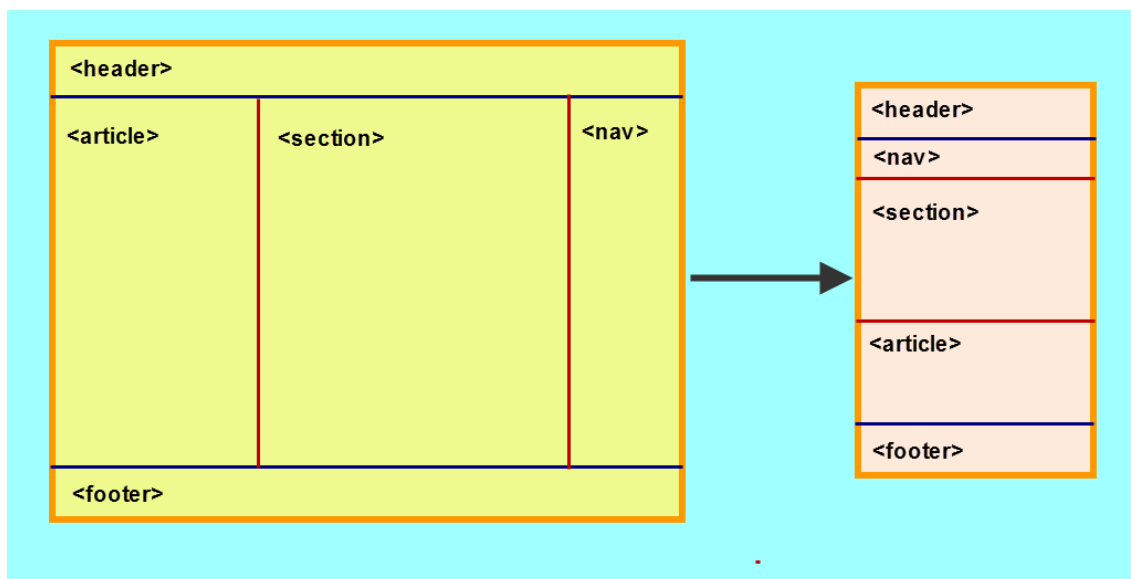


#### 4.11.4. Flex o Flexbox

La propiedad flex o Flexbox de CSS3 es la que permite que los elementos de un contenedor a la que se le ha aplicado esta propiedad se ajusten de la mejor manera posible al espacio disponible en el dispositivo en el que se visualiza, es decir, se aumenta el tamaño de los elementos para rellenar el espacio libre o se disminuye ese tamaño cuando esos elementos no caben en ese espacio.



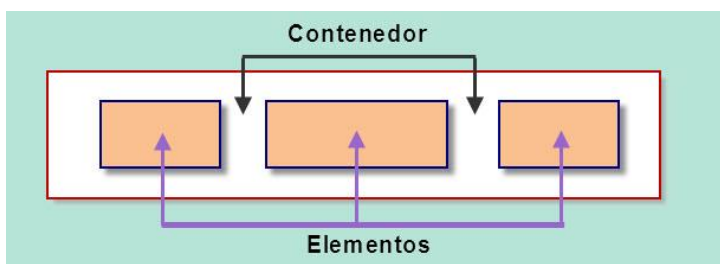
*La propiedad **Flexbox** permite visualizar elementos de una página web tanto en un pantalla normal como en un móvil simplemente cambiando su tamaño y su disposición sin tocar nada del código.*



En esta imagen se muestra como se vería un contenedor flexible con elementos también flexibles (<header>, <article>, etc.) de una página HTML5 en un dispositivo con una resolución concreta y como se vería en otro dispositivo con menor resolución. Se puede apreciar que el tamaño de los elementos varía en función del espacio que les ofrece el dispositivo y que el orden de visualización de dichos elementos también se puede cambiar.

Con esto se ve que Flexbox permite crear páginas cuyos elementos se adaptan perfectamente a la resolución de cualquier tipo de dispositivo existente sin tener que hacer modificaciones.

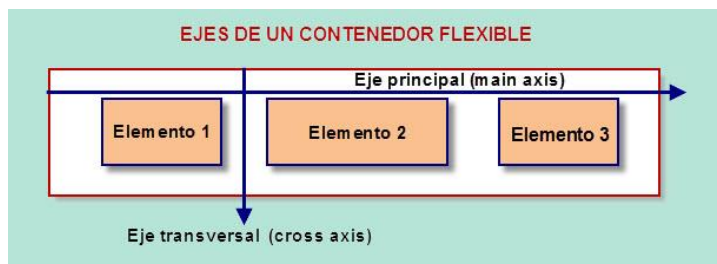
Las declaraciones de estilo CSS3 **display:flex;** o **display:inline-flex;** aplicadas a un elemento HTML lo convierten en un “contenedor flexible” y, de forma automática, los elementos que estén definidos dentro de él se convierten también en “elementos flexibles”.



	<i>No es conveniente usar <b>flexbox</b> para la distribución general de una página HTML5 sino que debe ser aplicada a sus componentes individualmente</i>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------

Un “contenedor flexible” tiene un eje principal (main axis) que marca la dirección en la cual se posicionan los elementos flexibles y otro eje

transversal (cross axis) perpendicular al anterior. A través de propiedades específicas aplicadas a esos ejes se podrán controlar la posición de cada elemento flexible con respecto a los demás.



Propiedad					
<b>flex</b>	11.0 10.0 - ms	29.0 21.0 -webkit-	28.0 18.0 -moz-	6.1 -webkit-	12.10

#### ○ **Propiedades del contenedor flexible**

- **display:** Define un contenedor flexible.

```
.contenedor { display: flex | inline-flex; }
```

Valores:

- *flex*: Tipo bloque.
- *inline-flex*: Tipo en línea.

- **flex-direction:** Establece la dirección en la que se colocarán los elementos flexibles del contenedor tomando como referencia el eje principal (main axis).

```
.contenedor { display :flex;
flex-direction: row | row-reverse | column | column-reverse;}
```

Valores:

- *row*: De izquierda a derecha (por defecto)
- *row-reverse*: De derecha a izquierda.
- *column*: De arriba abajo
- *column-reverse*: De abajo a arriba.

- **flex-wrap:** Indica si los elementos de un contenedor se colocan a lo largo de una o varias líneas tomando como referencia el eje principal (main axis)

```
.contenedor { display :flex;
 flex-wrap: nowrap | wrap | wrap-reverse; }
```

Valores:

- *nowrap*: Los elementos se colocan en una solo línea (por defecto)
  - *wrap*: Los elementos se disponen en varias líneas.
  - *wrap-reverse*: Los elementos se colocan en varias líneas pero en orden inverso.
- **flex-flow:** Sintaxis alternativa de las propiedades “flex-direction” y “flex-wrap” (por defecto valen *row* y *nowrap*).

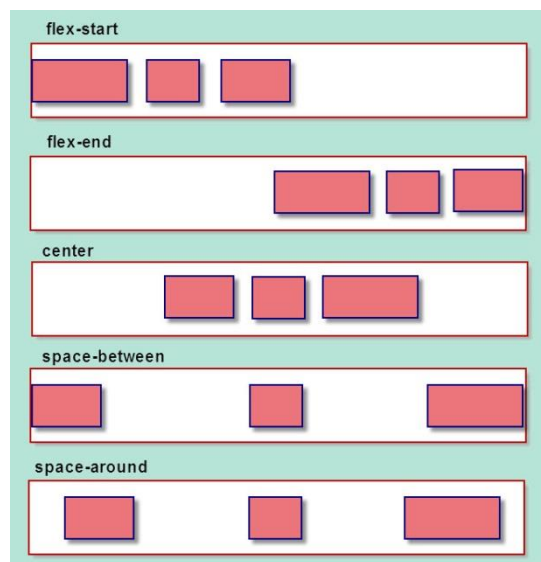
```
.contenedor { display :flex;
 flex-flow: Valor de flex-direction | Valor de flex-wrap; }
```

- **justify-content:** Define la alineación de los elementos a lo largo del eje principal (main axis).

```
.contenedor { display :flex;
 justify-content: flex-start | flex-end | center | space-between | space-around; }
```

Valores:

- *flex-start*: Los elementos se colocan pegados al borde izquierdo del contenedor (por defecto).
- *flex-end*: Los elementos se colocan pegados al borde derecho del contenedor.
- *center*: Los elementos se colocan centrados horizontalmente.
- *space-between*: Se coloca el primero junto al borde izquierdo y el último





junto al borde derecho dejando espacio entre los demás elementos.

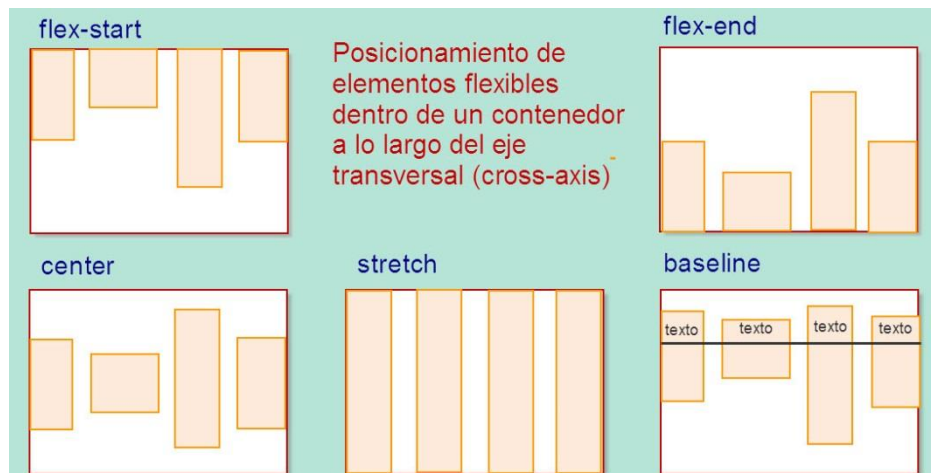
- *space-around*: Se colocan los elementos dejando separaciones iguales entre ellos.

- **align-items**: Define la alineación de los elementos a lo largo del eje transversal (cross axis).

```
.contenedor { display :flex;
align-items: flex-start | flex-end | center | stretch | baseline;}
```

Valores:

- *flex-start*: Los elementos se colocan pegados al borde superior del contenedor.
- *flex-end*: Los elementos se colocan pegados al borde inferior del contenedor.
- *center*: Los elementos se colocan centrados verticalmente.
- *stretch*: El ancho de los elementos se ajusta al ancho del contenedor (por defecto).
- *baseline*: Los elementos se ajustan verticalmente tomando como referente una determinada altura.



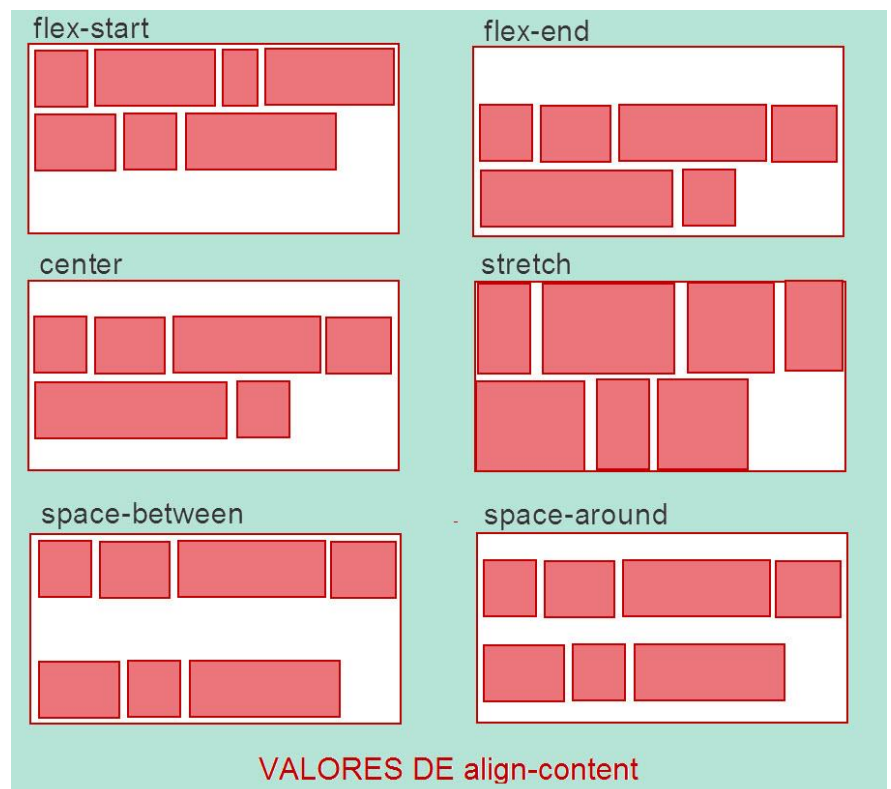
- **align-content**: Forma con todos los elementos del contenedor un bloque y luego define las alineaciones de dicho bloque con respecto al eje transversal (cross-axis).

```
.contenedor { display :flex;
align-content: flex-start | flex-end | center | space-between |
space-around | stretch;}
```

	<i>Solo se puede aplicar esta propiedad a los elementos flexibles tipo bloque que estén dentro del contenedor, nunca a elementos en línea.</i>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

Valores:

- *flex-start*: El bloque se posiciona junto al borde superior del contenedor.
- *flex-end*: El bloque se posiciona junto al borde inferior del contenedor.
- *center*: El bloque se coloca centrado verticalmente con respecto al contenedor.
- *space-between*: Se crea espacio entre las filas que forman los elementos del bloque.
- *space-around*: Se crea espacio alrededor de las filas que forman los elementos de bloque.
- *stretch*: Se ajustan las filas del bloque al espacio del contenedor (por defecto).



- **Propiedades de los elementos flexibles de un contenedor**
  - **order**: Indica el número de orden que se le aplicará a un elemento dentro de un contenedor, es decir, cuando se

cambie la resolución se puede cambiar el orden de visualización de los elementos de un contenedor con esta propiedad.

```
.elemento { order: número entero; }
```

Para el valor de **order** se pueden utilizar números enteros negativos (dichos elementos se visualizarían antes que los positivos) y, si no se especifica esta propiedad, se respeta el orden que tuvieran dentro del contenedor.

- **flex-grow:** Aumenta el tamaño de un elemento con respecto a los demás si es necesario.

```
.elemento { flex-grow: número; }
```

Donde *número* indica el número de veces que se aumenta con respecto a los demás elementos (0 no cambia el tamaño, 1 todos los elementos son iguales, 2 el elemento es doble que los demás, etc.).



*No se pueden utilizar números negativos*

- **flex-shrink:** Disminuye el tamaño de un elemento con respecto a los demás si es necesario.

```
.elemento { flex-shrink: número; }
```

Donde *número* indica el número de veces que se disminuye con respecto a los demás elementos (1 todos los elementos se dejan como están, 2 el elemento es la mitad que los demás, etc.).



*No se pueden utilizar número negativos*

- **flex-basis:** Determina la anchura inicial del elemento al que se aplica antes de distribuir el espacio libre de acuerdo a lo especificado tanto en flex-grow como en flex-shrink.

```
.elemento { flex-basis: anchura | auto; }
```

Si *anchura* vale 0, el espacio extra alrededor del elemento no se asigna. Si **flex-basis** vale “auto”, el espacio se distribuye a través del valor de flex-grow.

- **flex:** Sintaxis alternativa o shorthand de las propiedades flex-grow, flex-shrink y flex-basis en este orden. El segundo y tercer parámetro (flex-shrink y flex-basis) son opcionales. Los valores por defecto de **flex** es “0 1 auto”.

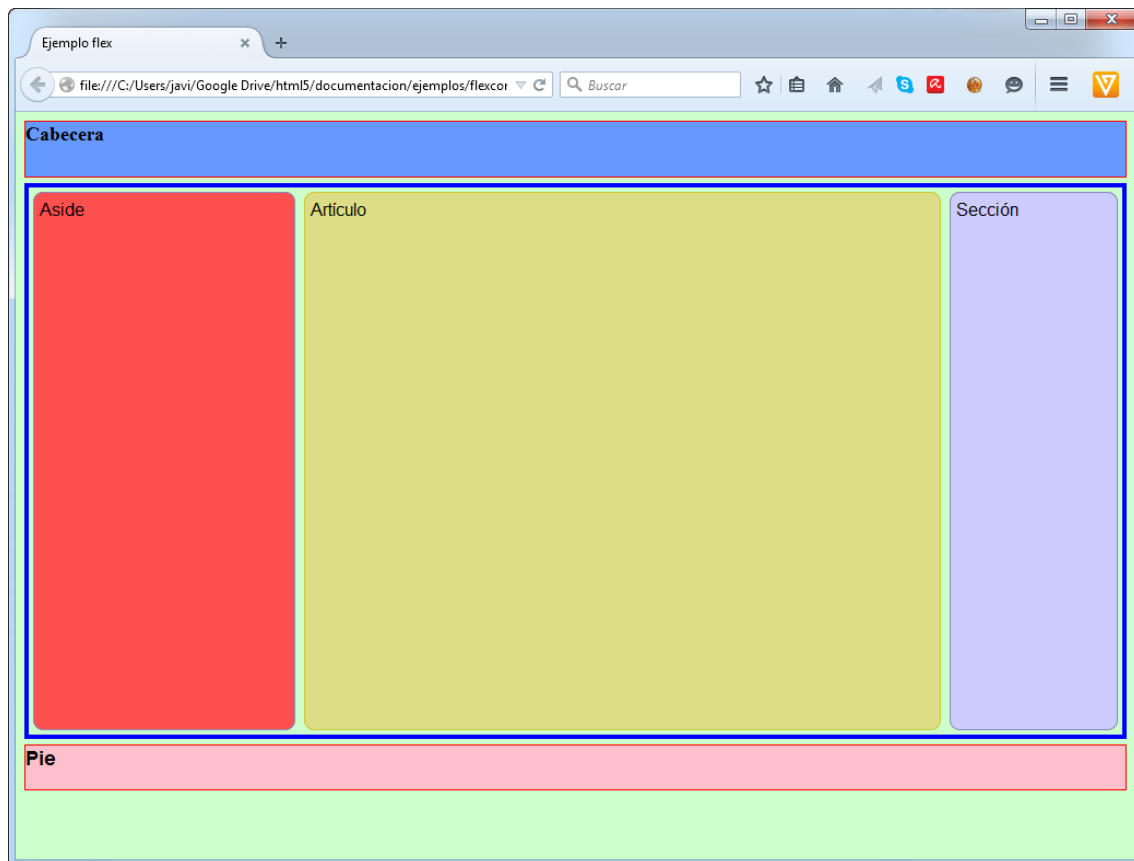
```
.elemento { flex: none | flex-grow flex-shrink flex-basis; }
```

- **align-self:** Esta propiedad determina la alineación por defecto del elemento al que se aplica esta propiedad.

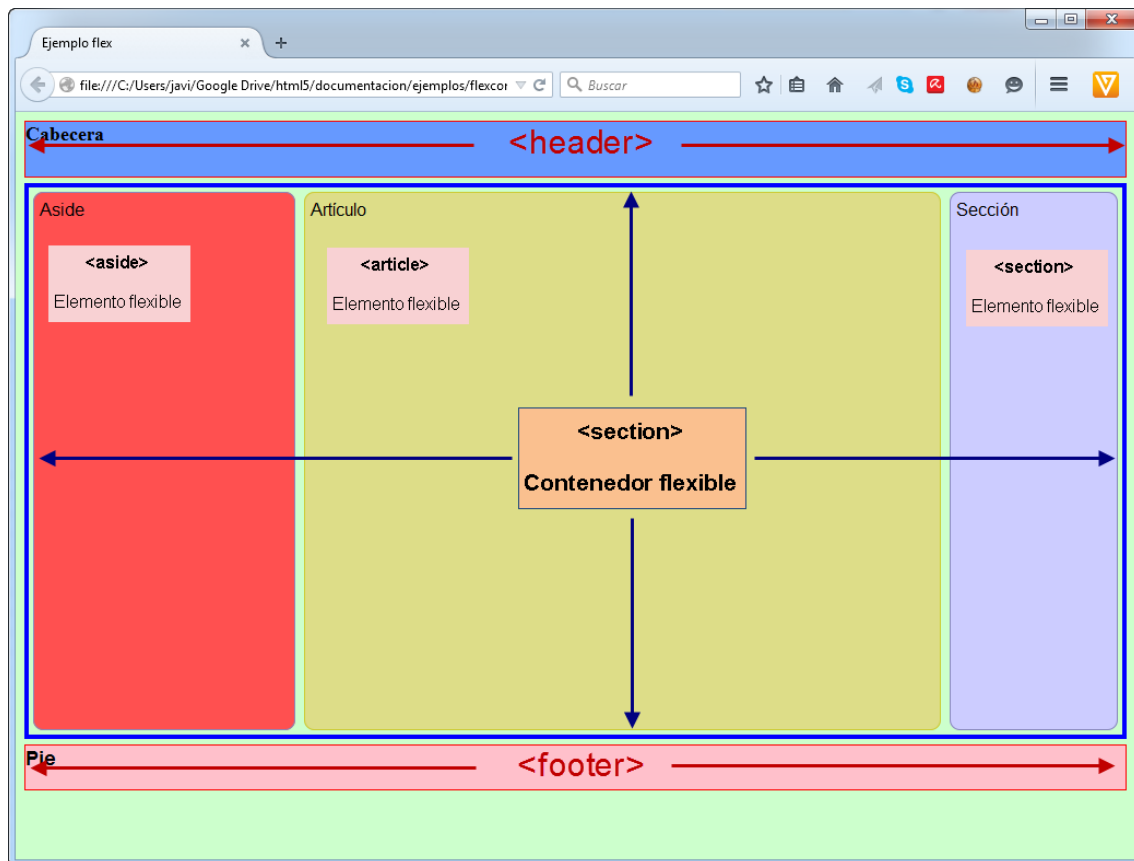
```
.elemento { align-self: auto | flex-start | flex-end | center |
baseline | stretch; }
```

Sus posibles valores están explicados en la propiedad **align-items**.

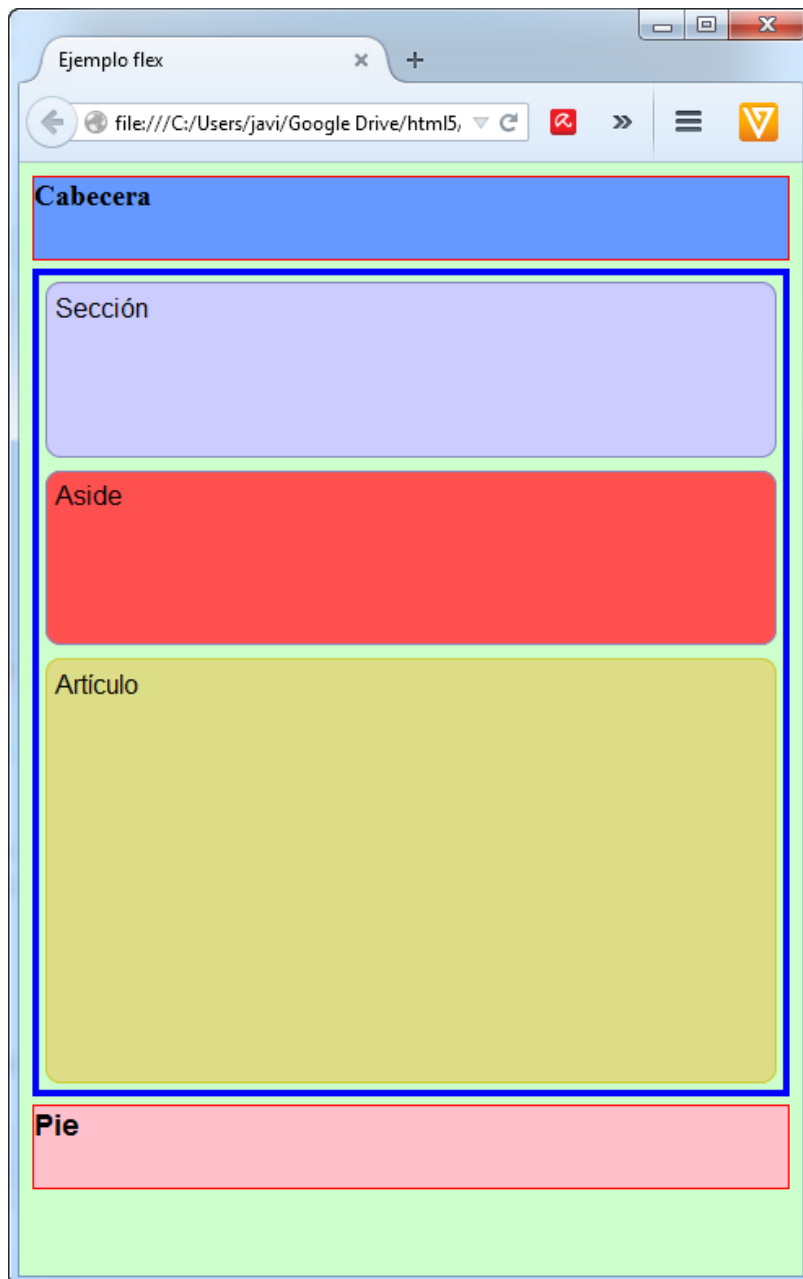
**Ejemplo:**



Tenemos esta página compuesta de un elemento `<header>` y un elemento `<footer>` no flexibles y luego un elemento `<section>` contenedor flexible de los elementos flexibles `<aside>`, `<article>` y `<section>`



Y lo que vamos a hacer es que, cuando se cambie a 640x480 pixels la resolución del dispositivo en el que se está visualizando la página, ésta cambie a la siguiente disposición



Como se ve en la imagen los elementos `<header>` y `<footer>` no cambian de lugar en la página ni de altura pero, en cambio, los elementos flexibles si que cambian de disposición al ponerse en columna con `<section>` primero, luego `<aside>` y, por último `<article>`, que ocupa más que los demás.

El código html completo de la página podría ser el siguiente:

```
<!DOCTYPE html>
<html lang="es-es">
<head>
```

```
<meta charset="UTF-8">

<title>Ejemplo flex</title>

<style type="text/css">

 body {background-color:#ccffcc;
 font: 16px Arial; }

 header {background-color: #6699FF;
 font: 18px Times New Roman;
 margin-bottom:5px;
 font-weight: bold;
 height: 50px;
 border: 1px solid red; }

 footer {background-color: pink;
 font: 18px Helvetica;
 font-weight: bold;
 height: 40px;
 margin-top: 5px;
 border: 1px solid red; }

 #contenedor-flexible {
 min-height: 500px;
 margin: 0px;
 padding: 0px;
```



```
 display: -webkit-flex;
 display: flex;
 -webkit-flex-flow: row;
 flex-flow: row;
 border: 4px solid blue; }

#contenedor-flexible>article {
 margin: 4px;
 padding: 5px;
 border: 1px solid #cccc33;
 border-radius: 7pt;
 background: #dddd88;
 -webkit-flex: 3 1 120px;
 flex: 3 1 120px;
 -webkit-order: 2;
 order: 2; }

#contenedor-flexible>section {
 margin: 4px;
 padding: 5px;
 border: 1px solid #8888bb;
 border-radius: 7pt;
 background: #ccccff;
 -webkit-flex: 1 6 80px;
 flex: 1 6 80px;
 -webkit-order: 3;
```

```
order: 3; }
```

```
#contenedor-flexible>aside {
 margin: 4px;
 padding: 5px;
 border: 1px solid #8888bb;
 border-radius: 7pt;
 background: #FF5050;
 -webkit-flex: 1 5 60px;
 flex: 1 5 60px;
 -webkit-order: 1;
 order: 1; }
```

```
@media all and (max-width: 640px) {
```

```
 #contenedor-flexible{
 -webkit-flex-flow: column;
 flex-flow: column; }
```

```
 #contenedor-flexible>article, #contenedor-flexible>section,
```

```
 #contenedor-flexible>aside {
 -webkit-order: 0;
 order: 0; }
```

```
 #contenedor-flexible>article, header, footer {
```

```
 min-height: 50px;
 max-height: 50px;
```

```

 }

}

</style>

</head>

<body>

 <header>Cabecera</header>

 <section id="contenedor-flexible">

 <article>

 Artículo

 </article>

 <section>

 Sección

 </section>

 <aside>

 Aside

 </aside>

 </section>

 <footer> Pie</footer>

</body>

</html>

```

Vamos a centrarnos ahora en el código tanto del contenedor flexible como de sus elementos:

- **Contenedor flexible <section> con id="contenedor-flexible"**

```

#contenedor-flexible {

 min-height: 500px;
}

```

```
margin: 0px;
padding: 0px;
display: -webkit-flex;
display: flex;
-webkit-flex-flow: row;
flex-flow: row;
border: 4px solid blue; }
```

En esta parte de la página se indica que el elemento con id="contenedor-flexible" será un contenedor flexible (display: -webkit-flex; y display: flex;) y que sus elementos se dispondrán en fila de izquierda a derecha (-webkit-flex-flow: row; flex-flow: row;).

- **Disposición general cuando se cambia la resolución**

```
@media all and (max-width: 640px) {
 #contenedor-flexible{
 -webkit-flex-flow: column;
 flex-flow: column; }

 #contenedor-flexible article, #contenedor-flexible section,
 #contenedor-flexible aside {
 -webkit-order: 0;
 order: 0; }

 #contenedor-flexible article, header, footer {
 min-height: 50px;
 max-height: 50px;
 }
```

```

 }
}

```

Cuando se cambie la resolución a 640 pixels de ancho en cualquier dispositivo (@media all and (max-width: 640px)), los elementos que contiene el elemento flexible “#contenedor-flexible” se pondrán en columna (-webkit-flex-flow: column; flex-flow: column;), o sea, de arriba abajo.

Los elementos <article>, <section> y <aside> del elemento flexible “contenedor-flexible” conservan el orden que tuvieran antes de cambiar la resolución (-webkit-order: 0; order: 0;)).

```

#contenedor-flexible article, #contenedor-flexible section,
#contenedor-flexible aside {
 -webkit-order: 0;
 order: 0;
}

```

Y, por último, que, cuando se cambie la resolución, los elementos <article> de “contenedor-flexible”, <header> y <footer> tengan una altura exacta de 50 pixels.

```

#contenedor-flexible article, header, footer {
 min-height: 50px;
 max-height: 50px;
}

```

- **Comportamiento del elemento flexible <article> incluido en “contenedor-flexible”**

La propiedad “flex” (-webkit-flex para los navegadores que lo necesiten) y “order” (-webkit-order para los navegadores que lo necesiten) marcan el comportamiento de este elemento

```

-webkit-flex: 3 1 120px;
flex: 3 1 120px;

```

```
-webkit-order: 2;

order: 2;
```

- “flex” es sintaxis alternativa de flex-grow, flex-shrink y flex-basis e indica que inicialmente tiene 120 pixels de ancho (flex-basis: 120px;) y que, si se aumenta la resolución, este elemento ocupa 3 veces más que los demás (flex-grow:3;) y que, si se disminuye la resolución, se queda como está (flex-shrink: 1;).
- “order” indica en qué posición se visualizará el elemento con respecto a los demás componentes del contenedor (en este caso order vale 2 por lo que será el segundo elemento del contenedor).

- **Comportamiento del elemento flexible <section> incluido en “contenedor-flexible”**

Al igual que el elemento anterior, las propiedades “flex” (--webkit-flex) y order (-webkit-order) son las que marcan el comportamiento de dicho elemento.

```
-webkit-flex: 1 6 80px;

flex: 1 6 80px;

-webkit-order: 3;

order: 3;
```

- “flex” indica que tiene un ancho inicial de 80 pixels y que se queda como está cuando se aumenta la resolución (flex-grow:1;) y disminuye 6 veces cuando se hace más pequeña.
- “order” indica que es el tercer elemento dentro del contenedor

- **Comportamiento del elemento flexible <aside> incluido en “contenedor-flexible”**

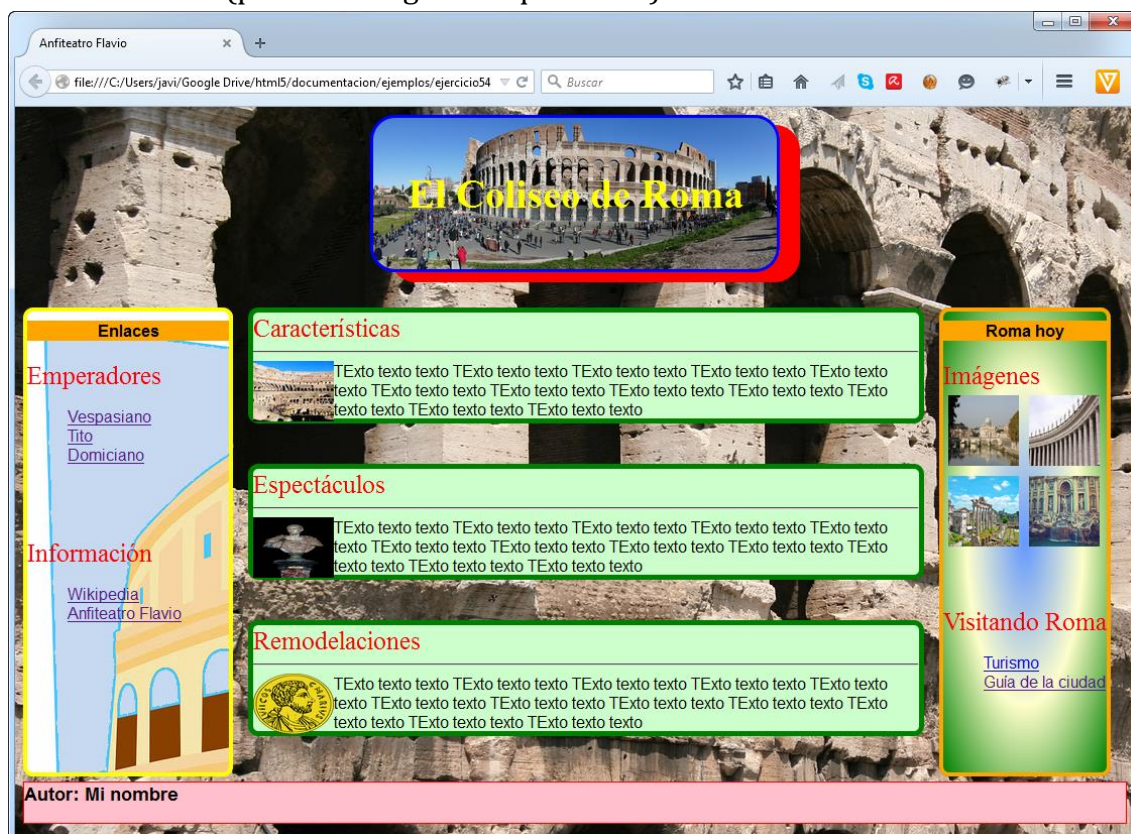
Al igual que los elementos anteriores, las propiedades “flex” (--webkit-flex) y order (-webkit-order) son las que marcan el comportamiento de este elemento.

```
-webkit-flex: 1 5 60px;
```

```
flex: 1 5 60px;
-webkit-order: 1;
order: 1;
```

- “flex” indica que tiene un ancho inicial de 60 pixels, que ocupa lo mismo cuando se aumenta resolución y que se disminuye 5 veces cuando se hace más pequeña.
- “order” indica que es el primer elemento dentro del contenedor.

**Ejercicio 54:** En este ejercicio vamos a practicar como bordes, fondos y gradientes. Para ello vamos a tomar como base un monumento como es el Coliseo o anfiteatro Flavio de Roma (puedes escoger cualquier otro).



Esta imagen se muestra lo que tienes que hacer. Respeta la estructura pero las características ornamentales como son colores, fondos, etc., ponlas a tu gusto.

El código HTML5 que puedes tomar como referencia es el siguiente con la imagen de cómo se visualizaría sin estilos y su distribución.

```
<!DOCTYPE html>

<html lang="es-es">
```

```
<head>

 <meta charset="UTF-8">

 <title>Anfiteatro Flavio</title>
</head>

<body>

 <header>El Coliseo o Anfiteatro Flavio </header>

 <aside>

 <header> Enlaces </header>

 Emperadores

 Vespasiano

 Tito

 Domiciano

 Información

 Wikipedia

 Anfiteatro Flavio

 </aside>
```



```
<section>

 <article>

 Características

 Texto Texto texto texto texto

 </article>

 <article>

 Espectáculos

 Texto texto texto

 </article>

 <article>

 Remodelaciones

 Texto texto texto

 </article>

</section>

<footer> Autor: Mi nombre</footer>

</body>
</html>
```

Anfiteatro Flavio
HTML Color Picker
CSS background prop...
Tryit Editor v2.3
Anfiteatro Flavio

file:///C:/Users/javi/Google Drive/html5/documentacion/ejemplos/ejerci
Buscar

El Coliseo de Roma
Enlaces
Emperadores

- Vespasiano
- Tito
- Domiciano

Información

- Wikipedia
- Anfiteatro Flavio

Características

IMAGEN

Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto

texto texto Texto texto texto Texto texto texto Texto texto texto

Espectáculos

IMAGEN

Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto

texto texto Texto texto texto Texto texto texto Texto texto texto

Remodelaciones

IMAGEN

Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto texto texto Texto texto texto

Texto texto texto Texto texto texto Texto texto texto Texto texto texto

Roma hoy

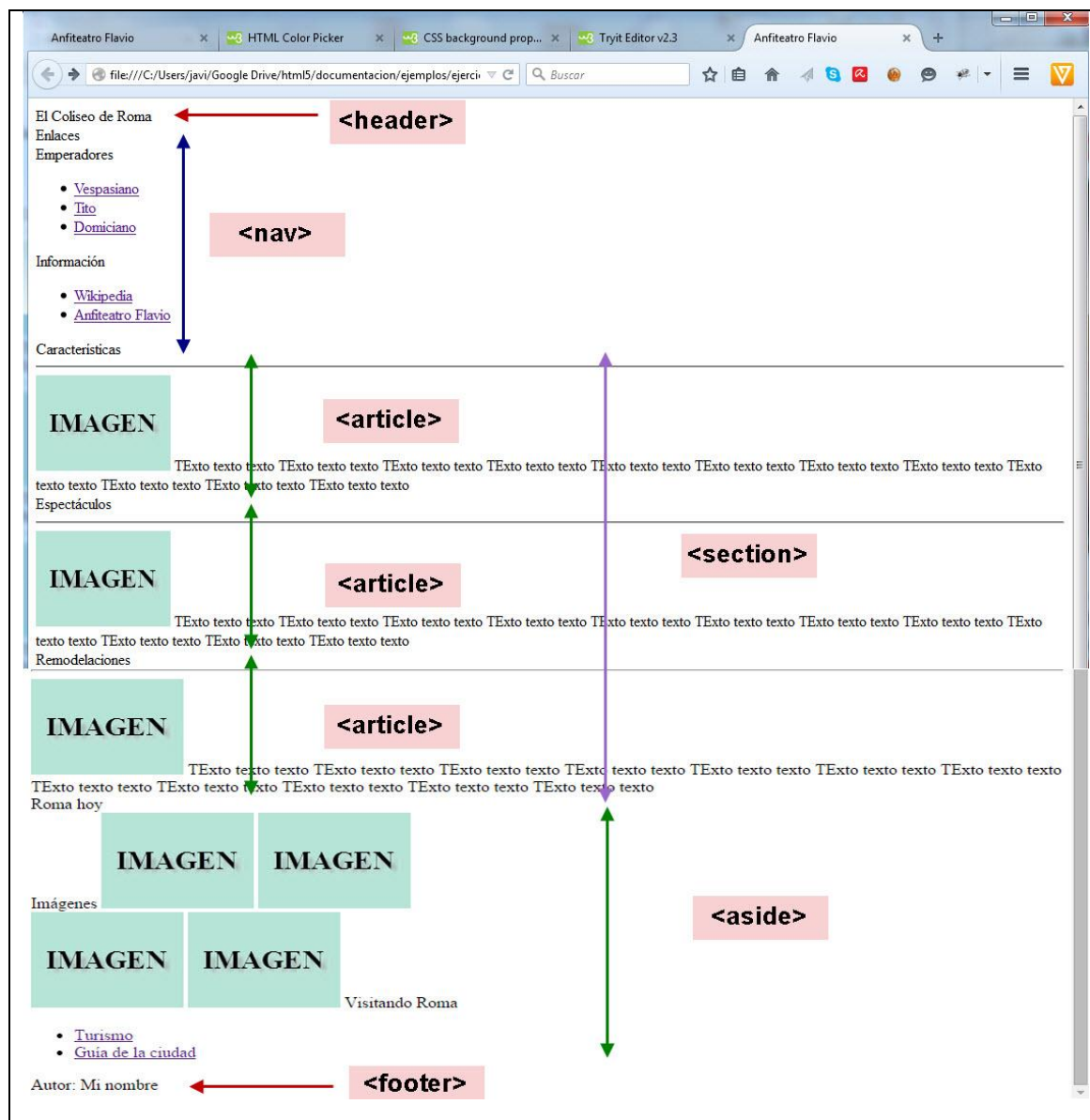
IMAGEN IMAGEN

Imágenes
IMAGEN IMAGEN

Visitando Roma

- Turismo
- Guía de la ciudad

Autor: Mi nombre



Los elementos tienen las siguientes características:

- **<body>**
  - Imagen de fondo (busca la que más te guste en Internet)
- **<header>**
  - Crea un estilo específico y único para este elemento con:
    - Fuente: Times New Roman de 40 pixels.
    - Anchura: 400 pixels.
    - Altura: 150 pixels.
    - Borde: 3 pixels, sólido y azul.
    - Redondeo del borde: 25 pixels
    - Sombra de la caja: 20 pixles, 10 pixels y roja.
    - Texto: Color amarillo, en negrita y centrado vertical y horizontalmente.

- Fondo: Será una imagen que busques en Internet y ajusta el tamaño del fondo al del elemento.
- Separación de los elementos inferiores: 35 pixels.
- **<nav>**
  - Altura: 100%.
  - Anchura: 200 pixels.
  - Fondo: Imagen que busques en Internet, que no se repita y se ponga en la posición superior izquierda.
  - Borde: 4 pixels, sólido y amarillo.
  - Redondeo borde: 10 pixels.
  - Visualización: La adecuada para ajustarse con los elementos <section> y <aside>
- **<section>**
  - Anchura: 700 pixels.
  - Altura: 100%
  - Visualización: La adecuada para ajustarse con los elementos <nav> y <aside>
  - Sin borde ni imagen ni color de fondo.
  - **<article>**
    - Color de fondo: #ccffcc
    - Margen superior: 0 pixels.
    - Margen inferior: 40 pixels.
    - Margen izquierdo: 15 pixels.
    - Margen derecho: 15 pixels
    - Borde: 5 pixels, sólido y verde.
    - Redondeo borde: 10 pixels.
    - **<img>**
      - Anchura: 80 pixels.
      - Altura: 60 pixels
      - Flotando a la izquierda con respecto al texto.
      - Escoge las imágenes que quieras de Internet.
- **<aside>**
  - Altura: 100%
  - Fondo: Gradiente radial con los colores #6699FF, #FFFFCC y verde.
  - Borde: 4 pixels, sólido y naranja.
  - Redondeo borde: 10 pixels.
  - Visualización: La adecuada para ajustarse con los elementos <nav> y <section>
  - **<img>**
    - Altura y anchura: 70 pixels.

- Espacio alrededor de todos los lados de la imagen: 5 pixels.
- Visualización: La adecuada para poner dos imágenes juntas.
- **<footer>**
  - Fondo de color: Rosa
  - Fuente: 18 pixels, Helvética.
  - Borde: 1 pixel, sólido y rojo.
  - Altura: 40 pixels.
  - Margen superior: 5 pixels.
  - Texto en negrita
- **<span>**
  - Visualización en bloque.
  - Texto en cursiva
  - Color del texto: Rojo.
  - Fuente: 25 pixels, Times.
- **<ul>**
  - Sin viñeta.
- Guarda la página como ejercicio54.html y crea la hoja de estilos ejercicio54.css para los estilos respetando las ubicaciones que llevamos durante todo el tema.



***Recuerda que una página web es algo personal por lo que no dudes en distribuir los elementos como quieras y poner los colores y fondos que más te gusten***

**Ejercicio 55:** Tenemos la siguiente página html con la que vamos a practicar las características de los elementos flexibles:

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <meta charset="UTF-8">

 <title>Flexbox</title>
```

</head>

<body>

<header>

Título

</header>

<article>

<p>

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

</p>

<p>

Segundo Párrafo Segundo Párrafo Segundo Párrafo

Segundo Párrafo Segundo Párrafo Segundo Párrafo

Segundo Párrafo Segundo Párrafo Segundo Párrafo

Segundo Párrafo Segundo Párrafo Segundo Párrafo

</p>

<p>

Tercer párrafo Tercer párrafo Tercer párrafo

Tercer párrafo Tercer párrafo Tercer párrafo

Tercer párrafo Tercer párrafo Tercer párrafo

</p>

</article>

```
<footer>
Pie de página
</footer>
</body>
</html>
```

Guarda la página siguiendo las pautas de los anteriores ejercicios como ejercicio55.html.

Como verás esta página está compuesta por un elemento <header>, otro elemento <article> con tres párrafos (<p>) y, por último, un elemento <footer>.

El elemento <article> va a ser el contenedor flexible y los elementos <p> van a ser los elementos flexibles.

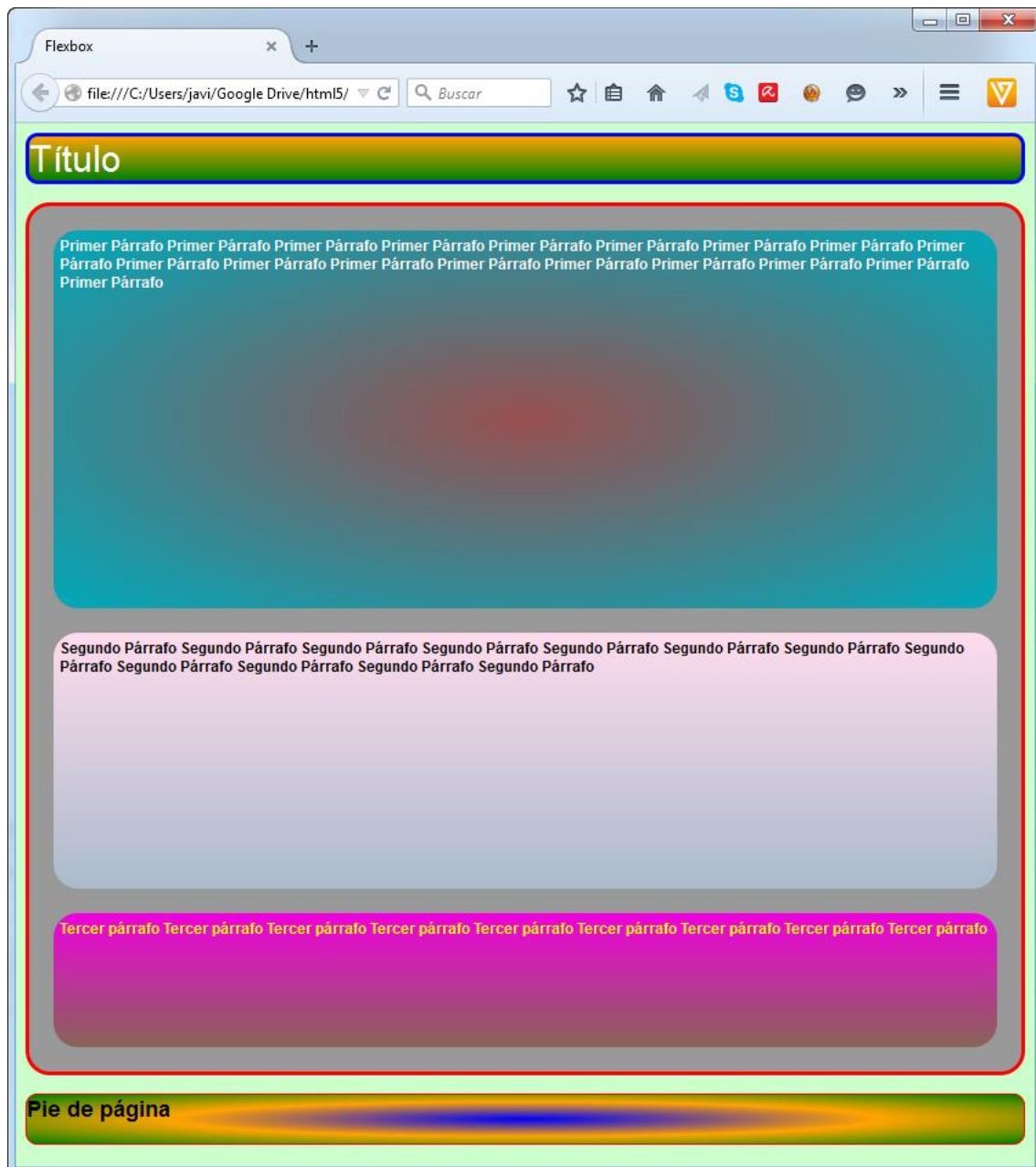
Antes de poner con las características flexibles, aplica los siguientes estilos a los elementos de la página guardándolos en el archivo flex.css.

- **<body>**
  - Color de fondo: #ccffcc
- **<header>**
  - Color de fondo: Gradiente lineal compuesto de naranja y verde.
  - Fuente: 30 pixels, Helvética.
  - Texto: Blanco en negrita
  - Borde: 3 pixels, sólido y azul.
  - Redondeo del borde: 10 pixels.
  - Margen inferior: 15 pixels.
- **<article>**
  - Crea un estilo específico para este elemento.
  - Anchura mínima: 500 pixels
  - Margen superior e inferior: 15 pixels.
  - Espacio entre el texto y el borde: 0 pixels.
  - Borde: 3 pixels, sólido y rojo.
  - Redondeo del borde: 20 pixels
  - Color de fondo: #999999
- **Clase “común” aplicable a los tres párrafos**
  - Margen: 20 pixels
  - Espacio entre el texto y el borde: 5 pixels

- Redondeo del borde: 20 pixels
- Fuente: 12 pixels Arial.
- Texto: Negrita.
- **Primer párrafo**
  - Color de fondo: Gradiente radial compuesto por los colores #995050 y #00aabb
  - Color de texto: Blanco.
  - Altura: 300 pixels.
- **Segundo párrafo**
  - Color de fondo: Gradiente lineal compuesto por los colores #ffddee y #aabbcc.
  - Color de texto: Negro
  - Altura: 200 pixels
- **Tercer párrafo**
  - Color de fondo: Gradiente lineal compuesto por los colores #f201e1 y #886655
  - Color de texto: Amarillo
  - Altura: 100 pixels
- **<footer>**
  - Color de fondo: Gradiente radial compuesto por los colores azul, naranja y verde.
  - Fuente: 18 pixels Helvética.
  - Texto: Negrita
  - Margen superior: 5 pixels
  - Borde: 1 pixel, sólido y rojo
  - Redondeo del borde: 10 pixels

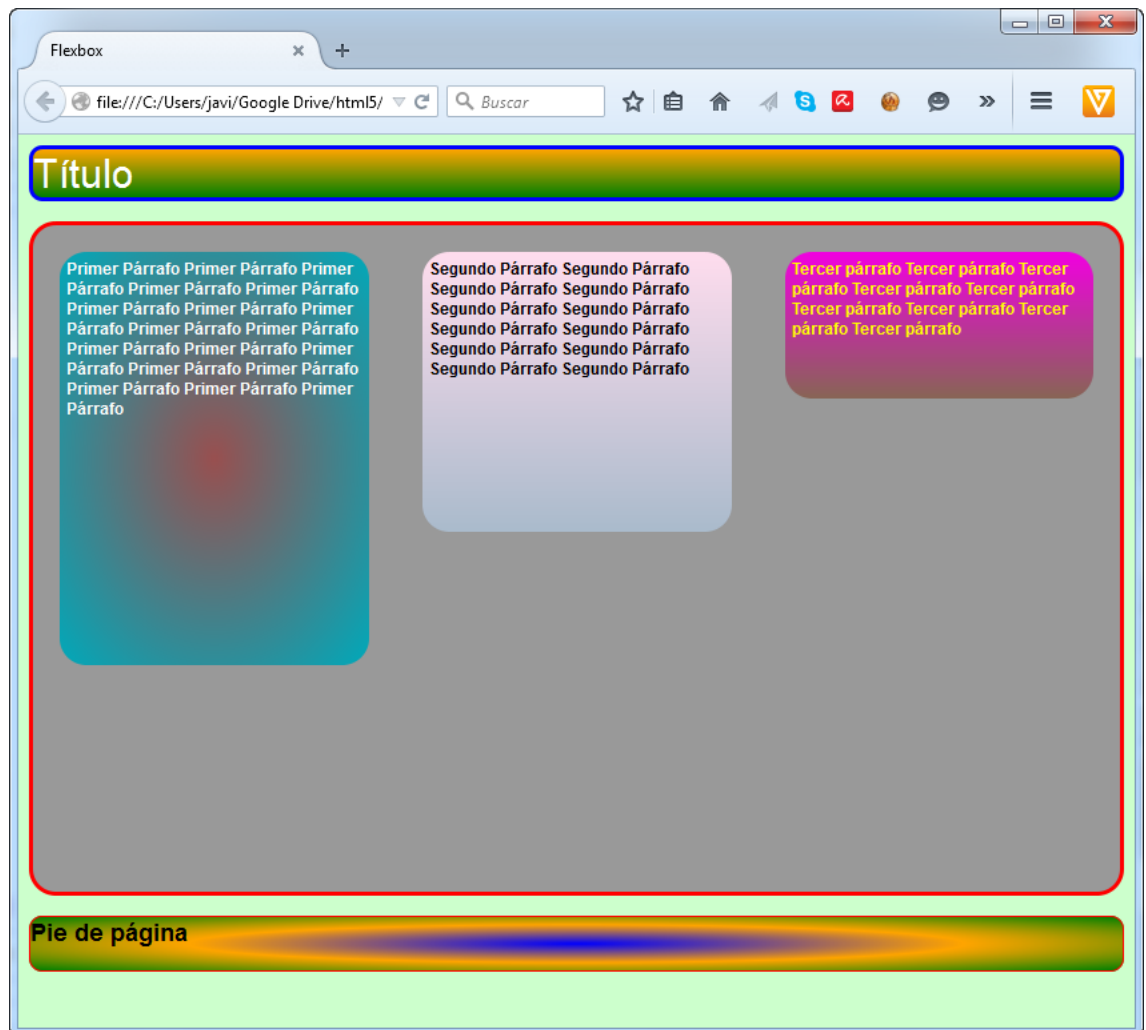
Aplicando los anteriores estilos, la página te tiene que quedar así.



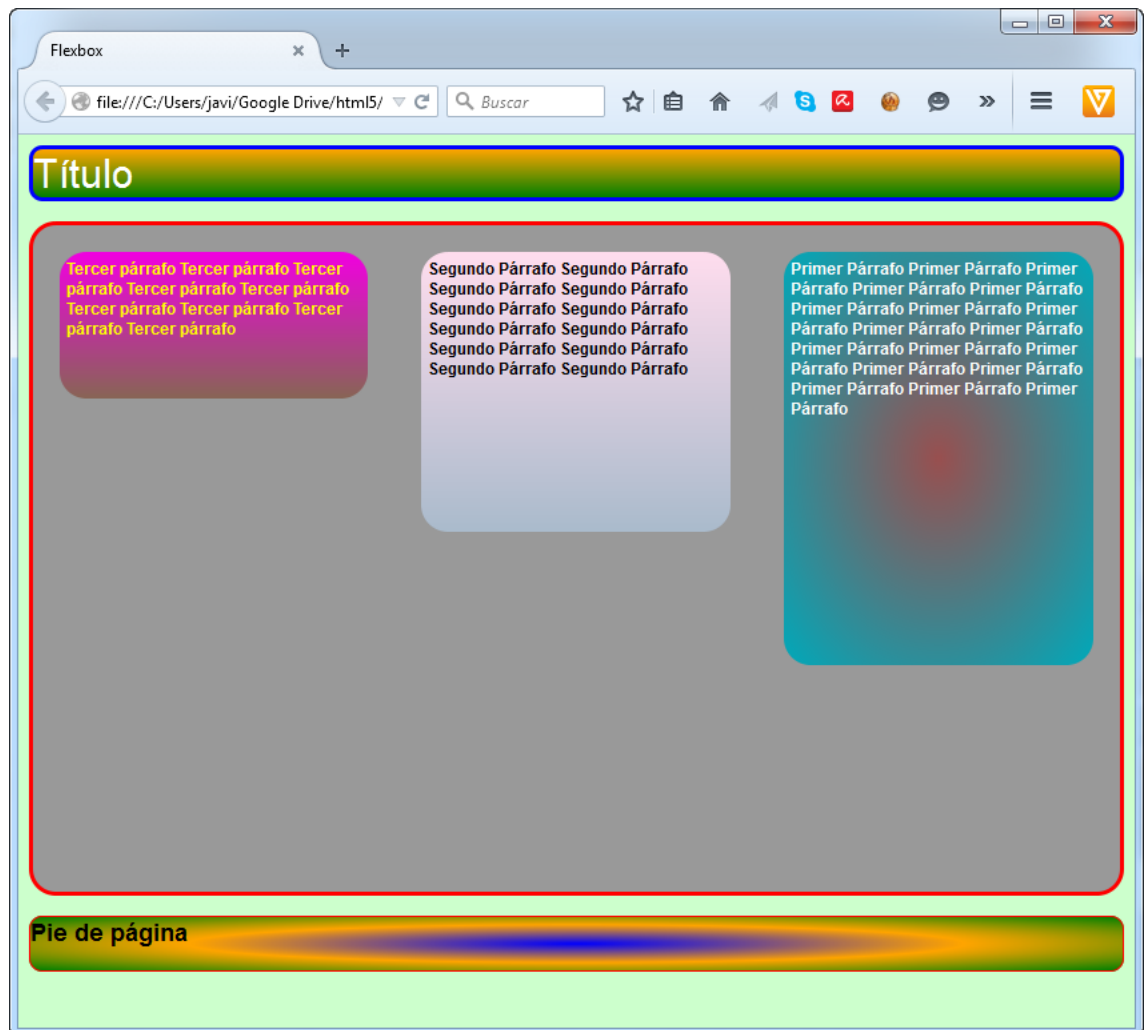


Vamos ahora a aplicar las características flexibles únicamente sobre el elemento `<article>`, o sea, el contenedor flexible, añadiéndolas/modificándolas al archivo `ejercicio55.css` (procura que valga para cualquier tipo de navegador aunque sea antiguo).

- **`<article>`**: Contenedor flexible
  - Que sus elementos se distribuyan de izquierda a derecha en una única fila.



- Que sus elementos se distribuyan de derecha a izquierda en una única fila



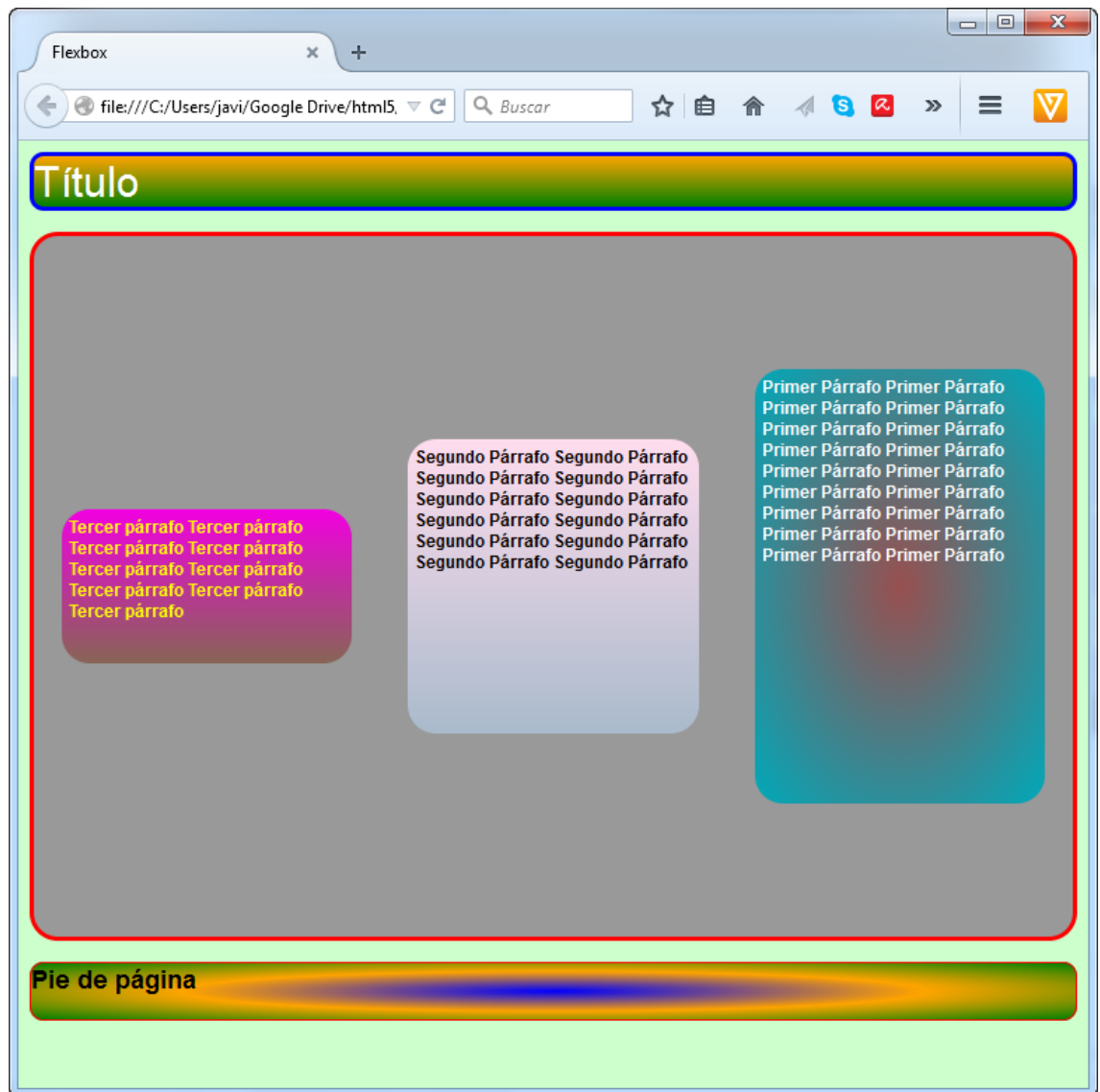
- Que sus elementos se distribuyan en una única columna de forma inversa



- Que sus elementos se distribuyan de izquierda a derecha en una única fila ajustados al borde inferior



- Que sus elementos se distribuyan de derecha a izquierda en una única fila centrados



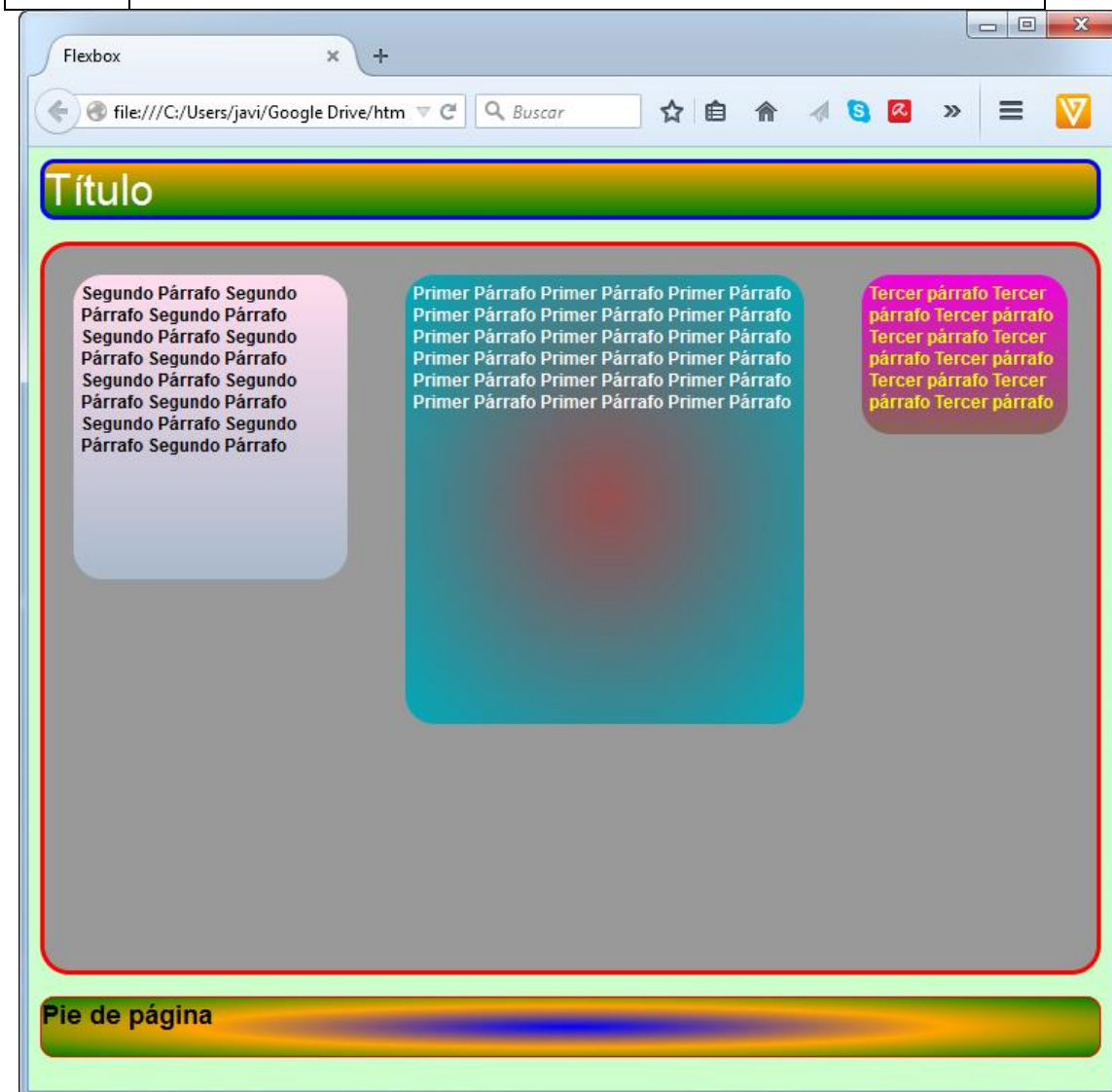
- **Elementos flexibles de <article>**

- Primer párrafo que ocupe el 50% de la anchura, sea el doble de los demás cuando se aumente la resolución, tres veces menos cuando se disminuya y su orden de visualización sea el segundo.
- Segundo párrafo que ocupe el 30% de la anchura, que no varía cuando se aumente la resolución, dos veces menos cuando se disminuya y su orden de visualización sea el tercero.
- Tercer párrafo que ocupe el 20% de la anchura, que triplique la de los demás cuando se aumente la resolución, sea igual a los demás cuando se disminuya y su orden de visualización sea el primero.
- Aumenta y disminuye la anchura del navegador para ver como influyen las condiciones anteriores.



*Recuerda que estos valores dependen de la distribución de los elementos y de si se puede aplicar esos cambios por lo que puede ser que, cuando cambies la resolución, no se modifiquen los elementos*

*como lo has indicado o que los contenidos no se ajusten correctamente a los párrafos.*



**Ejercicio 56:** Un contenedor flexible puede, a su vez, ser elemento flexible de otro contenedor.

Tenemos el siguiente código HTML5

```
!DOCTYPE html>

<html lang="es-es">

<head>

 <meta charset="UTF-8">

 <title>Flexbox</title>
```

```
</head>
```

```
<body>
```

```
<header>
```

```
Título
```

```
</header>
```

```
<article>
```

```
<p>
```

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

Primer Párrafo Primer Párrafo Primer Párrafo

```
</p>
```

```
<p>
```

Segundo Párrafo Segundo Párrafo Segundo Párrafo

Segundo Párrafo Segundo Párrafo Segundo Párrafo

Segundo Párrafo Segundo Párrafo Segundo Párrafo

Segundo Párrafo Segundo Párrafo Segundo Párrafo

```
</p>
```

```
<section>
```

```

```

```
<p>
```

Párrafo sección Párrafo sección Párrafo sección

Párrafo sección Párrafo sección Párrafo sección

Párrafo sección Párrafo sección Párrafo sección



```

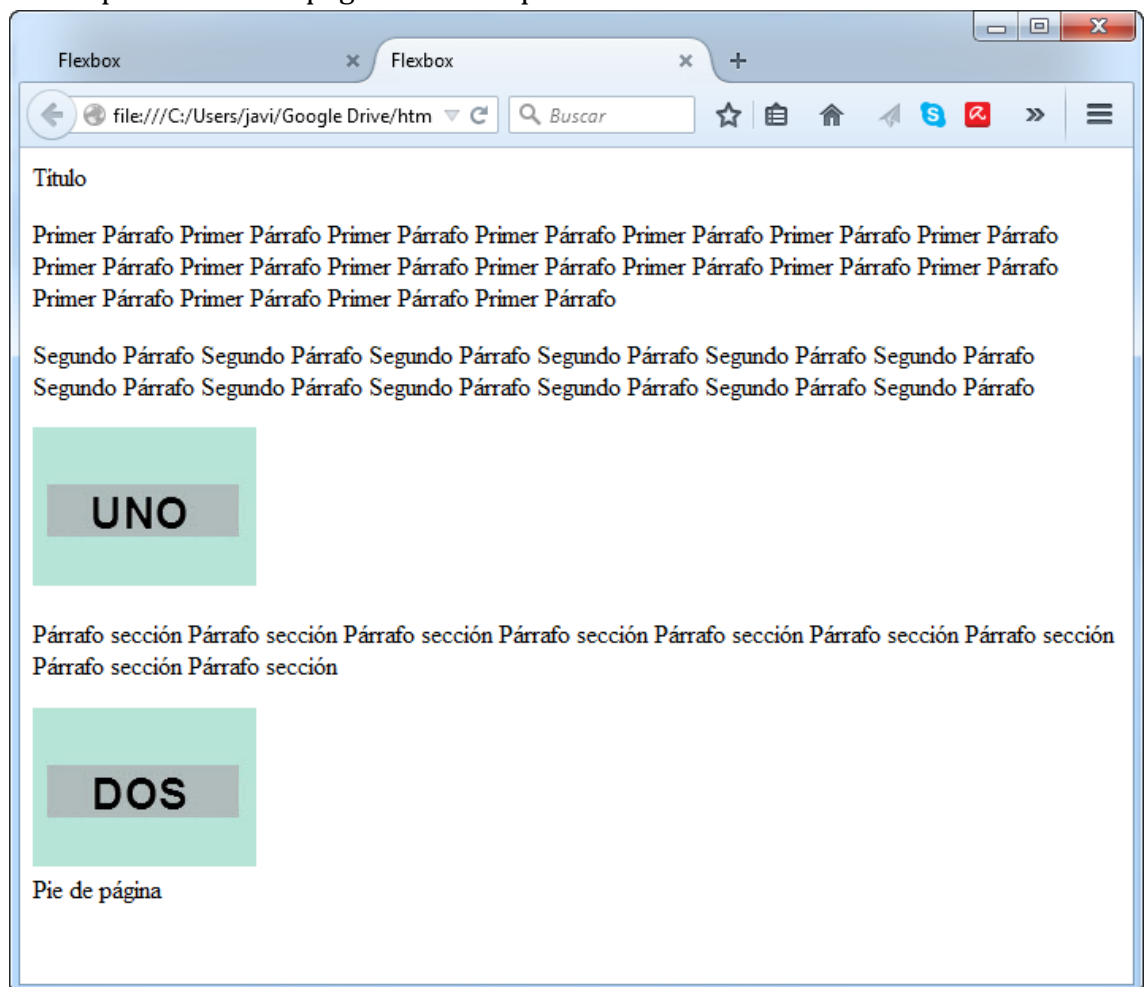
 </p>

 </section>
</article>

<footer>
 Pie de página
</footer>
</body>
</html>

```

Sin emplear estilos la página te debe quedar así



Como verás hay cambios con respecto al ejercicio anterior ya que el tercer párrafo ha sido eliminado y hay un nuevo elemento `<section>` dentro `<article>` formado por un párrafo `<p>` y dos elementos `<img>`.

Guarda el anterior código como ejercicio56.html cambiando las imágenes por dos que tengas a mano y ponles dimensiones relativamente pequeñas.

Vas a emplear el archivo flex.css creado para el ejercicio anterior y lo vas a agregar a ejercicio56.html con los cambios siguientes:

- **<article>**
  - Que sus elementos se distribuyan de izquierda a derecha en una única fila.
- **<section>** como elemento flexible de <article>
  - Crea un estilo específico para este elemento
  - Aplícale la clase que has creado para los párrafos.
  - Fondo: Gradiente lineal compuesto de los colores #112233 y #886655.
  - Altura: 100%.
  - Que ocupe el 20% de la anchura, que triplique la de los demás cuando se aumente la resolución, sea igual a los demás cuando se disminuya y su orden de visualización sea el primero.
- **Párrafo de <section>**
  - Fondo: Gradiente lineal compuesto de los colores #F201e1 #886655
  - Color de texto: Amarillo.
  - Altura: 60%.
  - Aplícale la clase común de los otros párrafos que has definido en el anterior ejercicio.

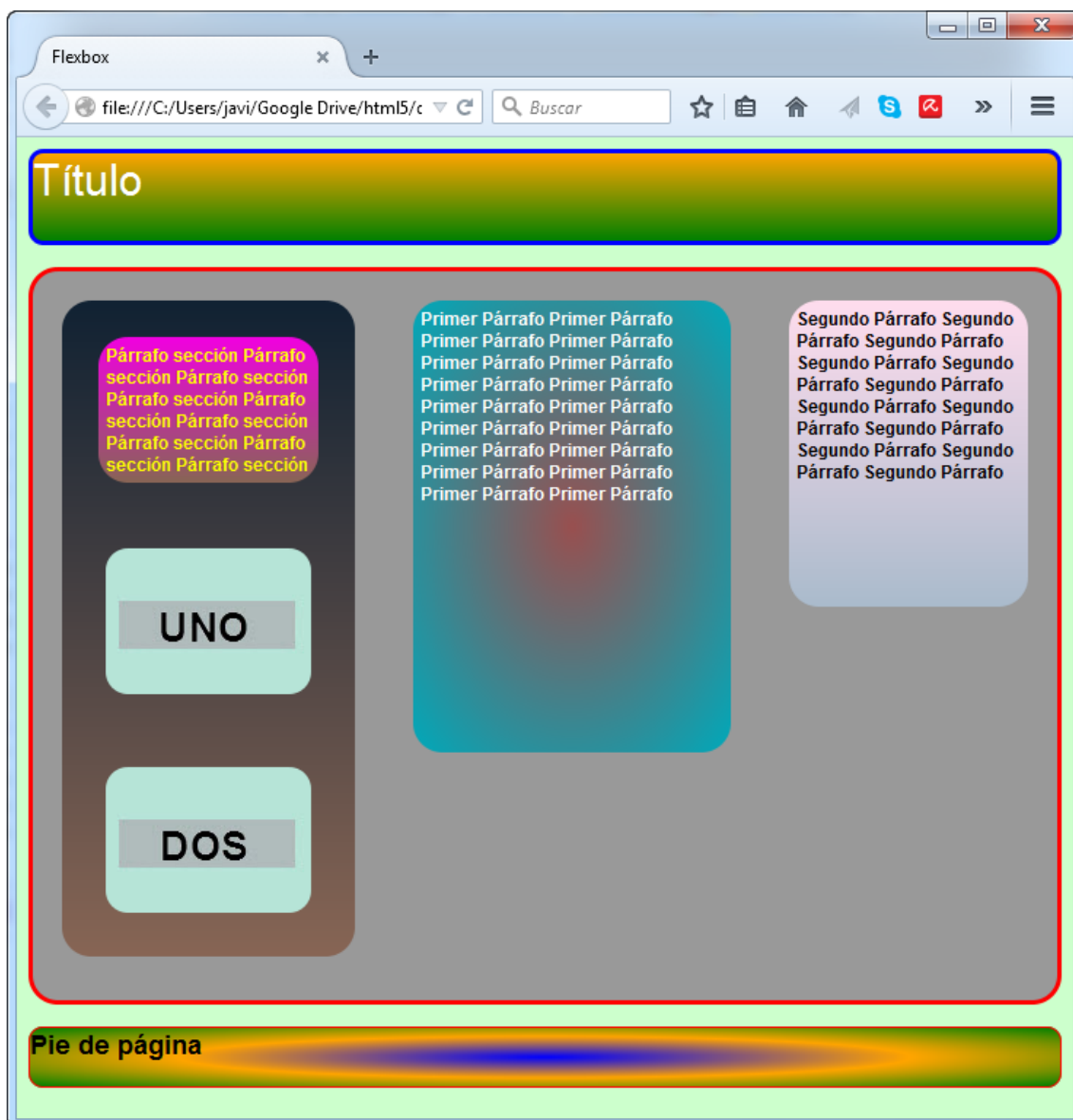
Después de aplicar los anteriores estilos junto con los ya definidos la página te tiene que quedar así:



Ahora vamos a hacer que `<section>` sea un contenedor flexible modificando los estilos que ya tenemos de la siguiente manera:

- **`<section>`**
  - Contenedor flexible que disponga sus elementos en una sola columna y en orden inverso.
  - **Primer elemento `<img>`**
    - Que ocupe el 25% de la anchura, que no varíe cuando se aumente la resolución, que sea igual a los demás cuando se disminuya y su orden de visualización sea el segundo.
  - **`<p>`**
    - Que ocupe el 50% de la anchura, que duplique la de los demás cuando se aumente la resolución, que no varíe cuando se disminuya y su orden de visualización sea el tercero.
  - **Segundo elemento `<img>`**
    - Que ocupe el 25% de la anchura, que triplique la de los demás cuando se aumente la resolución, sea igual a los

demás cuando se disminuya y su orden de visualización sea el primero.



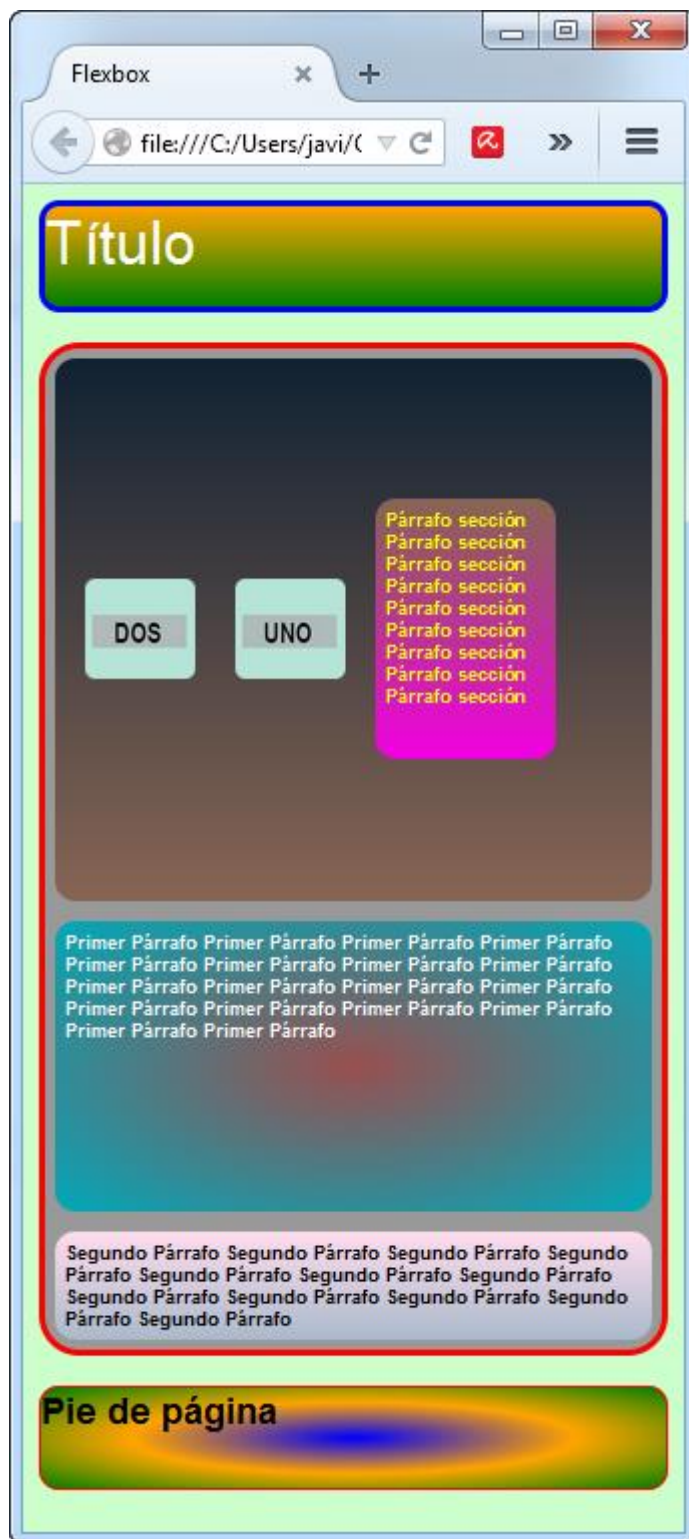
**Ejercicio 57:** En este ejercicio vamos a practicar cómo al cambiar la resolución de visualización de un dispositivo se puede modificar la disposición de los elementos e, incluso, su aspecto.

Agrega al archivo flex.css una parte que, cuando cambie la resolución a 480 pixels en cualquier dispositivo que se visualice la página, permita hacer lo siguiente:

- **<header>**
  - Altura: Exactamente igual a 50 pixels.
- **<footer>**
  - Altura: Exactamente igual a 50 pixels.

- **<article>** (recuerda que has definido un estilo para este elemento)
  - Que disponga sus elementos en una única columna.
- **<section>** (recuerda que has definido un estilo para este elemento)
  - Que disponga de sus elementos centrados en una sola fila.
  - **Los dos elementos <img>**
    - Anchura: 25 pixels.
    - Altura: 20 pixels.
  - **<p>**
    - Anchura: 80 pixels.
    - Altura: 120 pixels.
    - Color de fondo: Gradiente lineal con los colores #886655 y #F201e1, es decir, al revés de como los tenía.
- **Clase común a los párrafos**
  - Márgenes: 5 pixels.
  - Fuente: 9 pixels Arial.
  - Redondeo del borde: 10 pixels.
  - Texto en negrita.

Redimensiona tu navegador reduciendo su área y te tiene que quedar de la siguiente manera:



**Ejercicio 58:** Dependiendo de la resolución que tengas te puede haber pasado que el texto de los párrafos se salga de ellos haciendo un efecto bastante feo. Para resolverlo cambia la resolución que has definido en el ejercicio anterior por la que te cambie el aspecto antes de que se produzca lo anterior.

## **BLOQUE 5:**

## **HTML5**

## 5. HTML 5

### 5.1 Introducción

Las nuevas tendencias de la Web 2.0. invitan a que los usuarios tengan mucho más peso específico a la hora interactuar con los elementos informáticos que, antaño, solo estaban al alcance de los profesionales del tema. Pero, cuando el usuario es el centro de atención, se necesitan interfaces o entornos que le ofrezcan toda “la comodidad” posible a la hora de actuar con ellos, es decir, deben ser sencillos, muy intuitivos y prácticos además de cuidar los aspectos estéticos necesarios para potenciar su atractivo.

Un ejemplo claro de aplicación de lo expuesto en el párrafo anterior son los formularios HTML ya que permiten recoger información dentro de una página web rellenando campos o pulsando botones además de dar la posibilidad de enviarla a una dirección de correo o a un script dinámico escrito en php, asp o cgi.

El estándar HTML5 añade características nuevas para la realización de formularios mejorando sustancialmente los de las anteriores versiones HTML y presenta la novedad de la validación de datos por parte del navegador o del lado cliente o lo que simplifica en gran manera la creación y uso de formularios.

Los formularios en HTML5 no han cambiado mucho ya que mantiene la misma estructura pero se han agregado nuevos elementos, tipos de campo y atributos que les dan mayor potencia y mucha más sencillez a la hora de trabajar con ellos.

Un ejemplo que constata todo lo anterior es el elemento input el cual ha sido simplificado en su sintaxis pero ampliado en funcionalidades.

	<i>La especificación oficial del consorcio W3C sobre formularios se encuentra en la dirección:</i>   <b><a href="http://www.w3.org/TR/html5/forms.html">http://www.w3.org/TR/html5/forms.html</a></b>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

En la siguiente tabla tenemos los elementos HTML5 relacionados con los formularios

Elemento	Descripción
<form>	Define un formulario HTML
<input>	Define controles para el formulario
<textarea>	Define un área de texto
<label>	Define una etiqueta para un elemento <input>
<fieldset>	Grupos de elementos relacionados en un formulario
<legend>	Define un título para un elemento <fieldset>
<select>	Define una lista desplegable



<code>&lt;optgroup&gt;</code>	Define un grupo de opciones relacionadas en un lista desplegable
<code>&lt;option&gt;</code>	Define una opción en una lista desplegable
<code>&lt;button&gt;</code>	Define un botón
<code>&lt;datalist&gt;</code> 	Especifica una lista de opciones predefinidas para controles <code>&lt;input&gt;</code>
<code>&lt;keygen&gt;</code> 	Define un campo para generar pares de claves para sistemas de gestión de certificados
<code>&lt;output&gt;</code> 	Define el resultado de un cálculo

 → Nuevo en HTML5

## 5.2. REGLAS BÁSICAS

La recogida y envío de datos en un formulario se hace a través de elementos o controles como botones, listas desplegables, etc.

El *nombre de elemento* de cualquier componente de un formulario viene indicado por el atributo **name**.

Cada elemento tiene dos posibles valores: uno inicial y otro actual, que siempre son cadenas de caracteres. El valor inicial se asigna con el atributo **value** y el valor actual, que inicialmente tiene el valor inicial, es aquel que el usuario introduce bien escribiendo, marcando casillas o pulsando botones.

Cuando la información de un formulario se envía a un fichero indicado en el atributo **action** se hace a través de un tipo especial de botón denominado “submit”. Dependiendo de la forma de envío, especificada en el atributo **method**, la información de un formulario puede ir en la línea de llamada al archivo, o sea, en la barra de navegación (method get) o no ser mostrada (method post).

### 5.2.1. Tipos de controles

#### - Botón

Control que al ser pulsado o activado desencadena una acción. Los tipos de botones son:

- *Botón de envío (submit button)*: Cuando se pulsa o activa se envía la información del formulario.
- *Botón de reset (reset button)*: Cuando se pulsa o activa este botón se asignan los valores iniciales a todos los elementos del formulario.
- *Botón normal*: Cuando se pulsa o activa normalmente puede ejecutar uno o más scripts a través del atributo **event**.

#### - Casillas de verificación (checkbox)

Controles que permiten seleccionar cero o más opciones de un número limitado de ellas.

Una casilla de verificación está “seleccionada” cuando se activa el atributo **checked** correspondiente a la casilla.

También hay que tener en cuenta que, cuando se envía el formulario, solamente se manda la información de las casillas de validación seleccionadas y que varias casillas pueden tener el mismo nombre de control.

#### - Botones de opción (radio button)

Son controles que funcionan como las casillas de verificación a excepción de que, cuando se les asigna el mismo nombre de control, sólo puede estar uno de ellos activado.

#### - Menús

Son controles que ofrecen valores entre los que se puede elegir.

#### - Entradas de texto (text input)

Son controles que permiten introducir textos. Existen dos tipos:

- *De una sola línea:* A través del elemento **input** se crea este tipo de control.
- *De varias líneas:* A través del elemento **textarea** se crea este tipo de control.

En ambos casos el valor actual del control es el texto introducido.

- **Selección de ficheros (file select)**

Son controles que permiten escoger archivos de forma que la información de dichos ficheros pueda ser enviada con el formulario.

- **Controles ocultos (hidden controls)**

Son controles que no se ven pero cuyos valores actuales pueden ser enviados con el formulario. Normalmente el usuario que introduce datos en el formulario desconoce la existencia de estos controles y se suelen usar para pasar información adicional al script o programa que va a procesar la información del formulario.

- **Controles de tipo objeto (object controls)**

Son controles que pueden insertar objetos genéricos en los formularios de modo que los valores asociados se envíen junto con la información del formulario.

## 5.3. ELEMENTO <FORM>

Un formulario no es ni más ni menos que un conjunto de campos a través de los cuales el usuario introduce información que luego puede ser usada. Estos campos están representados por diferentes elementos de formulario como son los botones, casillas de chequeo, etc., que se adaptan a la forma de meter los datos y que suelen estar entre la etiqueta <form> y </form> del elemento **<form>**.

Etiqueta	<form>				
Descripción	Contiene los elementos de un formulario				
Elemento	Bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos				
Atributos propios	accept-charset, action, autocomplete, enctype, method, name novalidate, target				
Diferencia entre html 4.01 y html5	autocomplete y novalidate son atributos nuevos HTML5 y accept ha sido suprimido				

Atributos propios	Valor	Descripción
accept-charset 	Mapa de caracteres	Especifica un conjunto de mapas de caracteres que será usados para el envío del formulario. En HTML4.01 este conjunto estaba separado por comas.
action 	URL	Especifica la dirección a la que se envían los datos del formulario.
autocomplete 	on off	Especifica si el formulario debe ser completado (on) o no (off).

<b>enctype</b> 	application/x-www-form-urlencoded multipart/form-data text/plain	Especifica la codificación de envío del formulario (solo para método post)
<b>method</b> 	get post	Especifica el método de envío de los datos del formulario
<b>name</b> 	Texto	Especifica el nombre del formulario
<b>novalidate</b>  	novalidate	Especifica si se valida o no un formulario al enviarlo
<b>target</b> 	_blank _self _parent _top	Especifica dónde se va a visualizar el archivo que recibe los datos.

 → Nuevo en HTML5

La estructura más básica de un formulario es:

```
<form>
```

```
.
```

Elementos de formulario

```
.
```

```
</form>
```

Donde “Elementos de formulario” son controles que permiten recopilar información.

La forma anterior de usar un formulario no es muy práctica ya que normalmente los formularios sirven para que el usuario introduzca información y que ésta sea enviada a un script o fichero para ser procesada por lo que el elemento **<form>** debe tener algún atributo que permita ese envío. Así **<form>** se suele usar con los atributos method, action y enctype, siempre que method tenga valor "post", como mínimo.

```
<form action="URL">
```

```
.
```

```
Elementos de formulario
```

```
.
```

```
</form>
```

En la estructura anterior, el atributo method vale "get" (valor por defecto sino se pone) por lo que no se usa el atributo enctype.

```
<form action="URL" method="post" enctype="forma de codificación">
```

```
.
```

```
Elementos de formulario
```

```
.
```

```
</form>
```

En esta estructura si se usa el atributo enctype ya que method vale "post".

En los dos ejemplos anteriores "URL", o sea, el valor del atributo action es el lugar a donde vamos a enviar la información del formulario que puede ser o bien un script o programa php, javascript, etc., o una dirección de correo.

Cuando se envían los datos de un formulario a un script, éste debe estar construido para poder recogerlos correctamente y procesarlos. En cambio cuando se mandan a una dirección de correo se abre un cliente de correo y se envía un mensaje a la dirección especificada con el contenido del formulario.



*El desuso de los clientes de correo hace que en los formularios apenas se emplee esta forma de envío.*

Otra cuestión importante a tener en cuenta es saber qué manera de envío usar, es decir, cuando utilizar el atributo method con el valor "get" o "post".

- **get**

- Añade los datos del formulario a la URL del atributo action poniendo los nombres de los elementos más sus respectivos valores.
- La longitud de la URL con los datos anteriores es limitada (alrededor de 3000 caracteres).
- No se debe usar esta forma de envío cuando se manden datos confidenciales ya que son visibles en la URL.
- Útil para añadir a los marcadores de los navegadores la URL destino más los datos del formulario.
- Es mejor usar este valor para el envío de datos que no necesitan seguridad como las peticiones a los buscadores de información como Google.

### Ejemplo

```
<!DOCTYPE html>

<head>

 <title>Formulario</title>

 <meta charset="UTF-8">

</head>

<body>

 <form action="actualizar.php" method="get">

 <p>Nombre:<input type="text" name="Nombre" /></p>

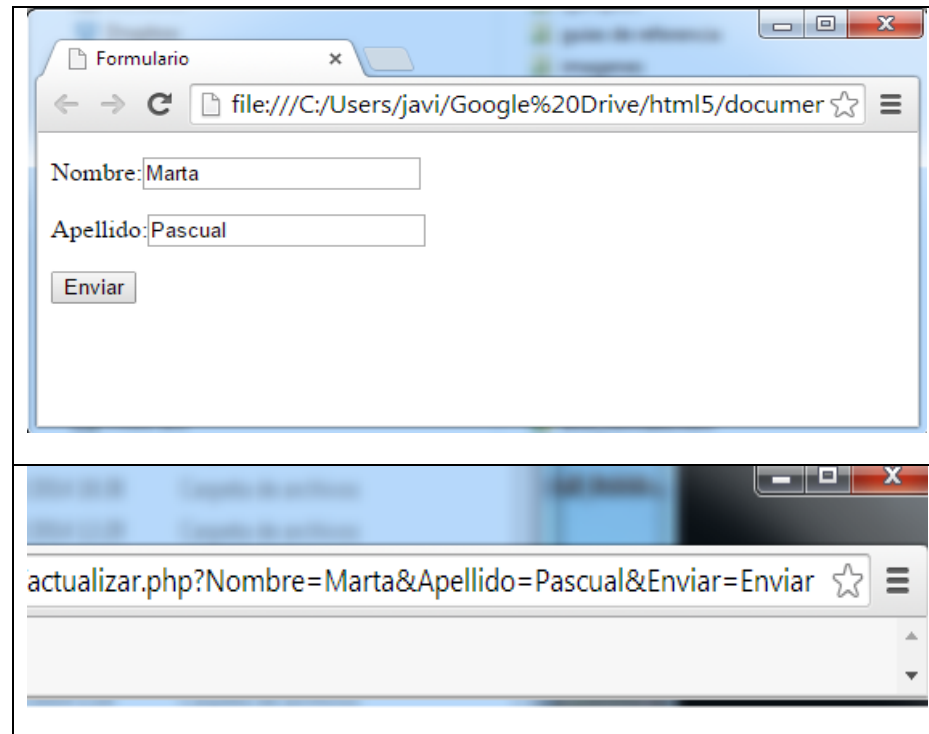
 <p>Apellido:<input type="text" name="Apellido" /></p>

 <p><input type="submit" name="Enviar"></p>

 </form>

</body>

</html>
```



En este ejemplo se define un formulario que envía datos al archivo “actualizar.php” (atributo action de <form>) que se encuentra en el mismo directorio que el documento html en el que está incluido dicho formulario, que usa method=”get”, que contiene dos cuadros de texto (elementos <input> con el atributo type=”text” y name=”nombre del elemento”) con nombres “Nombre” y “Apellido” y, también, un botón para enviar los datos al archivo indicado en “action”.

Como vemos en la primera imagen aparece el código html, en la segunda como se vería en un navegador (permite al usuario introducir un nombre y un apellido y enviarlo al documento actualizar.php al pulsar el botón “enviar”) y, en la última imagen, lo que aparece en la barra de direcciones del navegador después de pulsar el botón de enviar.

Al emplear method=”get” se ve que al valor de “action” se añade un ? y luego el nombre de los elementos igualados a su valor separados por un & y que, estos valores, son perfectamente visibles a la hora de cargar el archivo referido en “action” en el navegador.

#### - Post

- Inserta los datos del formulario dentro del cuerpo del envío por lo que no se añaden a la URL y, por lo tanto, no se ven.
- No hay limitación de caracteres a la hora de enviar datos.
- Los envíos con este valor no se añaden a los marcadores de los navegadores

#### Ejemplo;

```
<!DOCTYPE html>

<head>
```



```
<title>Formulario</title>

<meta charset="UTF-8">

</head>

<body>

 <form action="actualizar.php" method="post">

 <p>Nombre:<input type="text" name="Nombre" /></p>

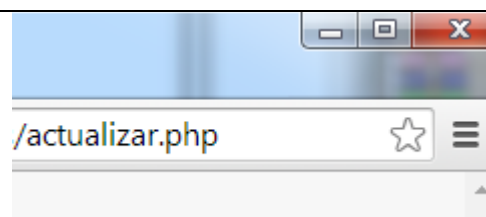
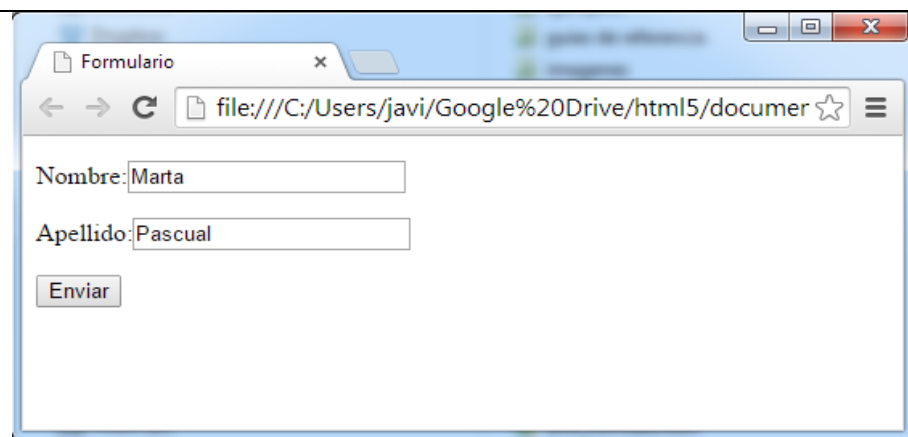
 <p>Apellido:<input type="text" name="Apellido" /></p>

 <p><input type="submit" name="Enviar"></p>

 </form>

</body>

</html>
```



Basándonos en el ejemplo que hemos usado explicando el valor **get** de “method” vemos que, al usar, el valor **post**, no se ve la información en la barra de direcciones al cargar el archivo actualizar.php.

### Ejemplos de <form>

```
<form action="recogerdatos.php" method="post"

 enctype="application/x-www-form-urlencoded">
```

```
.

Elementos de formulario

.

</form>
```

En este ejemplo se envían los datos al archivo “recogerdatos.php” que se encuentra en el mismo directorio que la página que contiene el formulario con la codificación “application/x-www-form-urlencoded” sin añadir ningún dato a la URL de “action” (“recogerdatos.php”) debido que a “method” tiene el valor post.

```
<form action="../recogerdatos.php" method="get">

.

Elementos de formulario

.

</form>
```

En este ejemplo el archivo receptor del formulario está un directorio por encima del documento html que lo contiene (action="../recogerdatos.php") y, al usar el valor get del atributo method, se verán los valores de los elementos del formulario en la barra de direcciones del navegador al cargar dicho archivo receptor.

```
<form action="http://www.mentor.mec.es/cgi-bin/validar.php"
method="post" enctype="application/x-www-form-urlencoded">

.

Elementos de formulario

.

</form>
```

La única diferencia con el ejemplo anterior es que en éste el archivo al que se envía el formulario está en la estructura del servidor http://www.mentor.mec.es.

```
<form action="mailto:uno@uno.es" method="post" enctype="text/plain">
```

Elementos de formulario

```
.
</form>
```

En este ejemplo al tener “action” el valor de una dirección de correo, se abre el cliente de correo que esté configurado en el sistema (Outlook express, gmail, etc.) y se envía un mensaje a la dirección de correo uno@uno.es con el contenido del formulario (nombres de elementos igualados a sus valores). Se utiliza method=”post” para no mostrar datos y se envían estos datos como texto plano.

```
<form>
```

```
.
```

Elementos de formulario

```
.
```

```
</form>
```

En este ejemplo al no tener asignado ningún valor inicial al atributo action significa que la URL de destino es el propio documento .html con method “get”. Normalmente se emplea esta forma cuando se envían los datos a un destino externo a través de script en vez de por el propio elemento <form>

```
<form>
```

```
.
```

Elementos de formulario

```
.
```

```
</form>
```

En este ejemplo se ve que no hay definido un valor para “action” por lo que el fichero destinatario del formulario es el propio documento html, no hay definido un valor para method por lo que se usa su valor por defecto que es get.

```
<form action="#">
```

```
.
```

## Elementos de formulario

</form>

En este ejemplo el valor “#” indica que el destino del formulario es una parte del propio documento html marcada por un ancla o enlace interno y con valor get para el atributo “method”.

### 5.4. Elemento <input>

Es el elemento más importante de los formularios ya que, a través de su atributo “type”, determina qué tipo de entrada o control es el que se le ofrece al usuario para que introduzca datos.

Etiqueta	<input>				
Descripción	Elemento para poner un control en un formulario				
Elemento	En línea y etiqueta vacía				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos				
Atributos propios	accept, alt, autocomplete, autofocus, checked, disabled, form, formaction, formenctype, formmethod, formnovalidate, formtarget, height, list, max, maxlength, min, multiple, name, pattern, placeholder, readonly, required, size, src, step, type, value y width				
Diferencia entre html 4.01 y html5	El atributo align ha sido suprimido en HTML5				

Los atributos comunes a todos los controles que se pueden crear con el elemento <input> están recogidos en la siguiente tabla

Atributos propios	Valor	Descripción
-------------------	-------	-------------

autocomplete      	on off	Especifica si un elemento <input> debe ser completado (on) o no (off)
autofocus      	autofocus	Especifica si un elemento <input> será el foco de la página cuando esta se carga
disabled      	disabled	Especifica que un elemento <input> está desactivado
form      	<i>Identificador formulario</i>	Especifica al formulario o formularios que pertenece el elemento <input>
max      	<i>número</i> <i>fecha</i>	Especifica el máximo valor del elemento <input>
maxlength      	<i>número</i>	Especifica el máximo número de caracteres permitidos en un elemento <input>
min      	<i>número</i> <i>fecha</i>	Especifica el mínimo valor del elemento <input>
multiple      	multiple	Especifica si el usuario puede introducir más de un valor
name      	<i>Texto</i>	Especifica el nombre del elemento <input>

pattern  	<i>Expresión</i>	Especifica una expresión contra la que se chequea el valor del elemento &lt;input&gt;
placeholder  	<i>Texto</i>	Especifica un breve descripción del valor del elemento &lt;input&gt;
readonly  	readonly	Especifica sin un campo de entrada es de solo lectura
required  	required	Especifica que un campo de entrada debe ser rellanado obligatoriamente antes de enviar el formulario
size  	<i>Número</i>	Especifica el ancho, en caracteres, de un elemento &lt;input&gt;
step  	<i>Número</i>	Especifica el número de intervalos para un elemento &lt;input&gt;
type  	button checkbox color date datetime datetime-local email file hidden image month number password radio range reset	Especifica el tipo de elemento &lt;input&gt;

	search submit tel text time url week	
value 	Texto	Especifica el valor del elemento <input>



→ Nuevo en HTML5

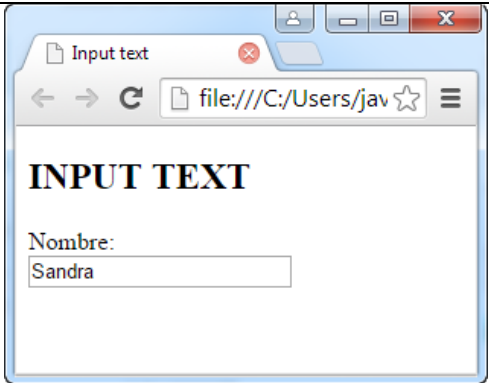
	<i>Los atributos <b>name</b> y <b>value</b> son los más importante de todos ya que sin ellos no se puede ni identificar un control ni enviar su valor.</i>
-----------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------

#### 5.4.1. Controles básicos <input>

##### - Cuadro de texto

Muestra un cuadro de texto para que el usuario escriba un contenido. Se inserta en un formulario con el valor "text" del atributo type.

```
<input type="text" name="nombre del control" value="valor inicial">
```

<pre>&lt;!DOCTYPE html&gt;  &lt;html lang="es-es"&gt;  &lt;head&gt;    &lt;title&gt;Input text &lt;/title&gt;    &lt;meta charset="UTF-8"&gt;  &lt;/head&gt;  &lt;body&gt;    &lt;h2&gt; INPUT TEXT &lt;/h2&gt;    &lt;form action=""&gt;</pre>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

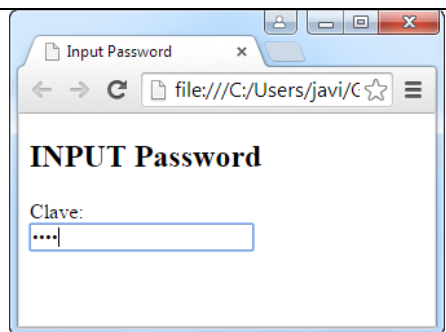
Nombre:   <input type="text" name="Nombre">   </form> </body> </html>	
--------------------------------------------------------------------------------------------	--

En este ejemplo el texto se introduce tal cual en el cuadro de texto.

#### - **Cuadro de contraseña**

Muestra un cuadro de texto en el que la información que introduce el usuario se muestra como un conjunto de asteriscos o círculos. Se inserta en un formulario con el valor "password" del atributo type.

```
<input type="password" name="nombre del control" value="valor inicial">
```

<!DOCTYPE html> <html lang="es-es"> <head> <title>Input Password </title> <meta charset="UTF-8"> </head> <body>   INPUT Password   <form action=""> Clave:   <input type="password" name="Clave">   </form> </body>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------



</html>	
---------	--

En este ejemplo el texto que se introduce en el cuadro de contraseña se convierte en “\*” pero solo a efectos de visualización ya que su valor es el que se escribe sin encriptar.

#### - Casilla de verificación

Muestra varias opciones de las que se puede elegir una, varias o ninguna de ellas. Se inserta en un formulario con el valor “checkbox” del atributo type.

```
<input type="checkbox" name="nombre control1" value="valor inicial1">
```

```
<input type="checkbox" name="nombre control2" value="valor inicial2">
```

```
<input type="checkbox" name="nombre control3" value="valor inicial3">
```

Si se quiere dejar una opción seleccionada se le añade el atributo “checked” sin ningún valor.

```
<input type="checkbox" name="nom" value="valor" checked>
```

```
<!DOCTYPE html>
```

```
<html lang="es-es">
```

```
<head>
```

```
<title>Input check </title>
```

```
<meta charset="UTF-8">
```

```
</head>
```

```
<body>
```

```
<h2> INPUT check </h2>
```

```
<form action="">
```

```
<input type="checkbox" name="comida"
value="si" >
```

```
 Comida

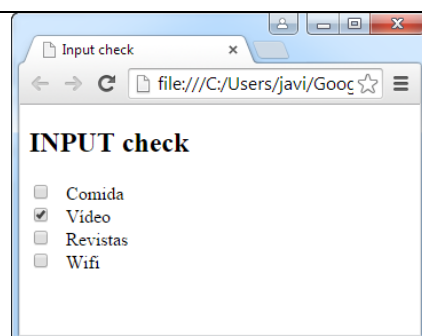
```

```
<input type="checkbox" name="video"
value="si" checked>
```

```
 Vídeo

```

```
<input type="checkbox" name="revistas"
```



<pre> value="si"&gt;  &amp;nbsp; Revistas&lt;br&gt;  &lt;input type="checkbox" name="wifi"  value="si"&gt;  &amp;nbsp; Wifi&lt;br&gt;  &lt;/form&gt;  &lt;/body&gt;  &lt;/html&gt; </pre>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

En este ejemplo los elementos marcados se envían o devuelven con el valor “si”, es decir, si se marca la casilla “Vídeo”, por ejemplo, entonces al elemento “video” se le asigna el valor “si”. Si se deja alguna casilla sin marcar entonces el elemento correspondiente no se envía ni devuelve ningún valor. También la casilla “Vídeo” estaría marcada por defecto al llevar el atributo “checked”.

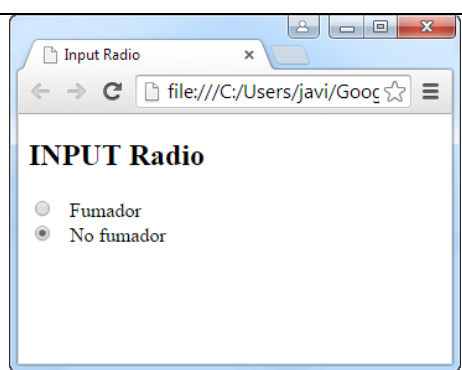
#### - Botón de opción

Muestra varias opciones de las que sólo se puede escoger una de ellas. Se inserta en un formulario con el valor “radio” del atributo type y deben tener el mismo valor en el atributo name.

```

<input type="radio" name="nombre" value="valor1" checked">
<input type="radio" name="nombre" value="valor2">
<input type="radio" name="nombre" value="valor3">

```

<pre> &lt;!DOCTYPE html&gt;  &lt;html lang="es-es"&gt;    &lt;head&gt;      &lt;title&gt;Input Radio &lt;/title&gt;      &lt;meta charset="UTF-8"&gt;    &lt;/head&gt;    &lt;body&gt;      &lt;h2&gt; INPUT Radio &lt;/h2&gt;      &lt;form action=""&gt;        &lt;input type="radio" name="fumador" </pre>	
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

<pre> value="si" &gt;      &amp;nbsp; Fumador&lt;br&gt; &lt;input type="radio" name="fumador" value="no" checked&gt;      &amp;nbsp; No fumador&lt;br&gt; &lt;/form&gt;  &lt;/body&gt;  &lt;/html&gt; </pre>	
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

En este ejemplo el control se llama de la misma manera (fumador) y se envía con el valor “si” o “no” dependiendo de la casilla que haya sido activada (hay que tener en cuenta que sólo puede haber una casilla activa). La casilla correspondiente a “no fumador” es que está marcada por defecto al llevar el parámetro “checked”.

#### - Botón de envío

Muestra un botón que, cuando se pulsa, envía el formulario al archivo indicado en la URL del atributo “action”. Se inserta en un formulario con el valor “submit” del atributo type.

```
<input type="submit" name="nombre" value="enviar">
```

<pre> &lt;!DOCTYPE html&gt;  &lt;html lang="es-es"&gt;    &lt;head&gt;      &lt;title&gt;Input Radio &lt;/title&gt;      &lt;meta charset="UTF-8"&gt;    &lt;/head&gt;    &lt;body&gt;      &lt;h2&gt; INPUT Radio &lt;/h2&gt;      &lt;form action="archivo.html"&gt;        &lt;input type="radio" name="fumador"          value="si" &gt;          &amp;nbsp; Fumador&lt;br&gt; </pre>	
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

<pre> &lt;input type="radio" name="fumador" value="no" checked&gt;  &amp;nbsp; No fumador&lt;br&gt;  &lt;input type="submit" value="enviar"&gt;  &lt;/form&gt;  &lt;/body&gt;  &lt;/html&gt; </pre>	
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

En este ejemplo, al pulsar el botón de enviar, mandaría los datos del formulario al fichero “archivo.html” el elemento “fumador” con el valor “si” o “no” dependiendo de la opción marcada.

#### - Botón de reset

Muestra un botón que, cuando se pulsa, les asigna el valor inicial a los elementos del formulario. Se inserta en un formulario con el valor “reset” del atributo type.

```
<input type="reset" name="nombre" value="Reset">
```

<pre> &lt;!DOCTYPE html&gt;  &lt;html lang="es-es"&gt;  &lt;head&gt;  &lt;title&gt;Input Radio &lt;/title&gt;  &lt;meta charset="UTF-8"&gt;  &lt;/head&gt;  &lt;body&gt;  &lt;h2&gt; INPUT Radio &lt;/h2&gt;  &lt;form action="archivo.html"&gt;  &lt;input type="radio" name="fumador" value="si" &gt;  &amp;nbsp; Fumador&lt;br&gt;  &lt;input type="radio" name="fumador" value="no" checked&gt; </pre>	
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------

<pre> &amp;nbsp; No fumador&lt;br&gt; &lt;input type="submit" value="enviar"&gt; &lt;input type="reset" value="Borrar"&gt; &lt;/form&gt; &lt;/body&gt; &lt;/html&gt; </pre>	
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

En este ejemplo se agrega otro botón de tipo “reset” que borrará los datos que hubiera en el formulario y pondría los que están por defecto.

#### - Botón

Muestra un botón que, cuando se produce algún evento sobre él, como ser pulsado, realiza alguna acción. Se inserta en un formulario con el valor “button” del atributo type.

```
<input type="button" onclick="alert('Pulsado') value="Pulsa el botón">
```

En este ejemplo se muestra el mensaje *Pulsado* con se pulsa el botón.

#### 5.4.2. Controles nuevos HTML5 <input>

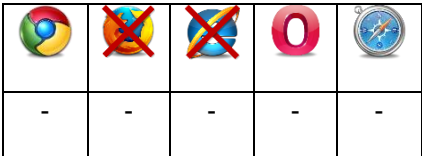
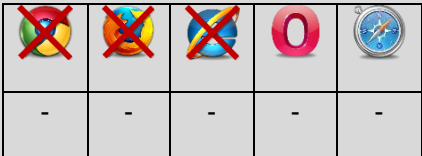
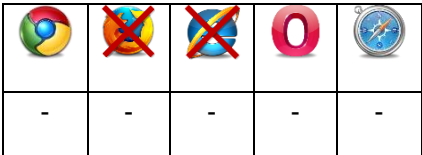
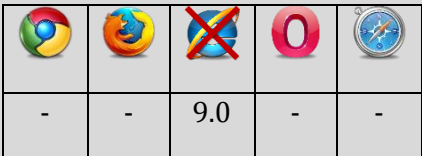
En la siguiente tabla tenemos los nuevos controles de formulario que se pueden crear con el elemento <input> en HTML5.

Los navegadores antiguos y algunos más modernos no soportan estos nuevos controles por lo que los interpretarán como un simple cuadro de texto.

También el aspecto de estos controles depende mucho del navegador variando de unos a otros.

Control	type	Explicación/ejemplo/navegadores que lo soportan								
Número	number	Contiene un valor numérico								
		<input type="number" name="número" value="1">								
		<table><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td>-</td><td>-</td><td>9.0</td><td>-</td><td>-</td></tr></table>						-	-	9.0
										
-	-	9.0	-	-						
Fecha	date	Contiene una fecha								

		&lt;input type="date" name="fecha" value="2014-11-24"&gt;                    -  -  -  -  -
Color	color	Contiene la identificación de un color   &lt;input type="color" name="color" value="#00ff00"&gt;   &lt;input type="color" name="color" value="red"&gt;                    -  -  -  -  -
Rango	range	Contiene un valor numérico comprendido entre un rango   &lt;input type="range" name="rango" min="1" max="10" value="7"&gt;                    -  -  9.0  -  -
Mes y año	month	Contiene un mes y un año   &lt;input type="month" name="mes" value="2014-03"&gt;                    -  -  -  -  -
Semana y año	week	Contiene un número de semana y un año   &lt;input type="week" name="semanayaño" value="2014-W13"&gt;               

			-	-	-	-	-	
Hora	time	Contiene una hora						
		<input type="time" name="hora" value="22:55">						
								
Fecha y hora	datetime	Contiene una fecha y una hora con zona horaria						
		<input type="datetime" name="fechayhora">						
								
Fecha y hora local	datetime-local	Contiene una fecha y una hora sin zona horaria						
		<input type="datetime-local" name="local">						
								
Correo	email	Contiene una dirección de correo						
		<input type="email" name="correo" value="uno@uno.es">						
								
Búsqueda	search	Contiene un término para campos de búsqueda						
		<input type="search" name="buscar" value="gatos">						
								

Teléfono	tel	Contiene un número de teléfono								
		<input type="tel" name="teléfono">								
		<table><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td>-</td><td>-</td><td>.</td><td>-</td><td>8.0</td></tr></table>						-	-	.
-	-	.	-	8.0						
Dirección Internet	url	Contiene una dirección de Internet								
		<input type="url" name="dirección" value="http://google.es">								
		<table><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td>-</td><td>-</td><td>9.0</td><td>-</td><td>-</td></tr></table>						-	-	9.0
-	-	9.0	-	-						

**Nota:** El símbolo “-” significa cualquier versión del navegador y si hay un número indica la versión a partir de la cual se reconoce el control.

## Ejemplos completos

### 5.5. Otros elementos de formulario

Aunque el elemento `<input>` permite crear una gran variedad de controles de formulario, existen otros tales como listas desplegables, áreas de texto, generadores de claves, etc., que usan otros elementos diferentes.

#### - `<select>`

Elemento que crea una lista desplegable cuyas opciones vienen definidas por elementos `<option>` que van dentro de las etiquetas de apertura y cierre de `<select>`.

Etiqueta	<code>&lt;select&gt;</code>				
Descripción	Elemento que crea una lista desplegable				
Elemento	En línea				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos				
Atributos propios	autofocus, disabled, form, multiple, name, required y size				
Diferencia entre html	autofocus, form y required son atributos HTML5 (*)				



## 4.01 y html5

(\*) Estos atributos están explicados en el apartado del elemento <input>

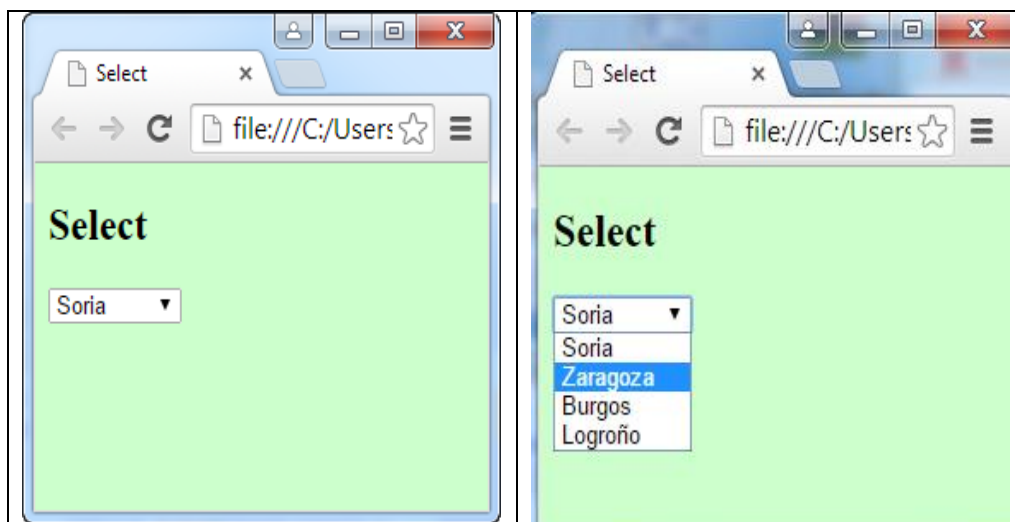
### - <option>

Elemento que forma parte de un elemento <select> o <datalist> y define sus posibles opciones y qué valores iniciales tendrán.

Etiqueta	<option>				
Descripción	Define las opciones de un elemento <select> o <datalist>				
Elemento	En línea				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos				
Atributos propios	disabled, label, selected y value				
Diferencia entre html 4.01 y html5	Ninguna				

### Ejemplo de <select> y <option>

```
<!DOCTYPE html>
<html lang="es-es">
 <head>
 <title>Estilos en tabla</title>
 <meta charset="UTF-8">
 <style type="text/css">
 .fondoverde
 {background-color:#ccffcc;}
 </style>
 </head>
 <body class="fondoverde">
 <h2> Select </h2>
 <form>
 <select name="origen">
 <option value="Soria" selected>Soria</option>
 <option value="Zaragoza">Zaragoza</option>
 <option value="Burgos">Burgos</option>
 <option value="Logroño">Logroño</option>
 </select>
 </form>
 </body>
</html>
```



En este ejemplo tenemos una serie de ciudades de destino de las cuales “Soria” está seleccionada y sería su valor por defecto. En cuanto se pulsa con el ratón sobre la flecha de la lista, ésta se despliega y deja elegir otro valor como sería en la segunda imagen “Zaragoza”.

#### - <textarea>

Elemento que define un cuadro de texto de n-columnas por n-filas

Etiqueta	<textarea>					
Descripción	Define un cuadro de texto					
Elemento	En línea					
Navegadores que la soportan	    					
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y de eventos					
Atributos propios	autofocus, cols, disabled, form, maxlength, name, placeholder, required, rows y wrap					
Diferencia entre html 4.01 y html5	autofocus, form, maxlength, placeholder, required, wrap son atributos HTML5					

#### Ejemplo:

```

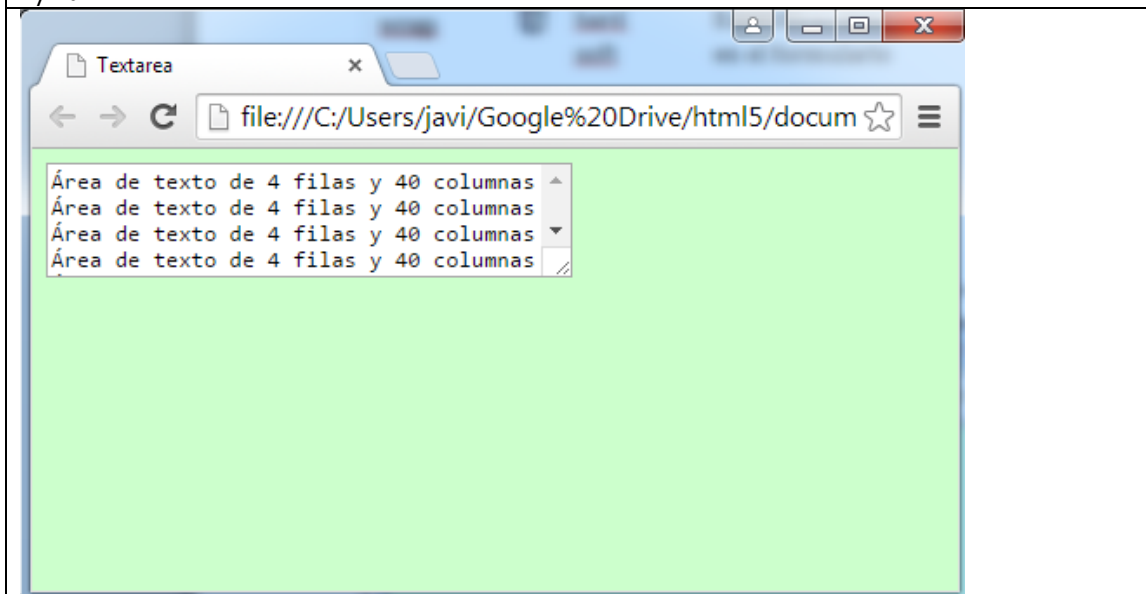
<!DOCTYPE html>
<html lang="es-es">
 <head>
<title>Textarea</title>
<meta charset="UTF-8">
<style type="text/css">
 .fondoverde
 {background-color:#ccffcc;
 }
</style>
</head>

```

```

<body class="fondoverde">
<textarea rows="4" cols="40">
Área de texto de 4 filas y 40 columnas
Área de texto de 4 filas y 40 columnas
Área de texto de 4 filas y 40 columnas
Área de texto de 4 filas y 40 columnas
Área de texto de 4 filas y 40 columnas
Área de texto de 4 filas y 40 columnas
Área de texto de 4 filas y 40 columnas
</textarea>
</body>
</html>

```



Como se ve en el ejemplo, el elemento “textarea” ha generado un área de texto donde sólo se ven 4 filas (para acceder a las demás habría que hacer scrolling) y 40 columnas.

#### - <button>

Define un botón sobre el cual se puede pulsar con el ratón.

Etiqueta	<button>				
Descripción	Define un botón para ser pulsado con el ratón				
Elemento	En línea				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y de eventos				
Atributos propios	autofocus, disabled, form, fromaction, formenctype, formmethod, formnovalidate, formtarget, name, type, value				
Diferencia entre html 4.01 y html5	autofocus, form, fromaction, formmethod, formnovalidate, formtarget son atributos HTML5				

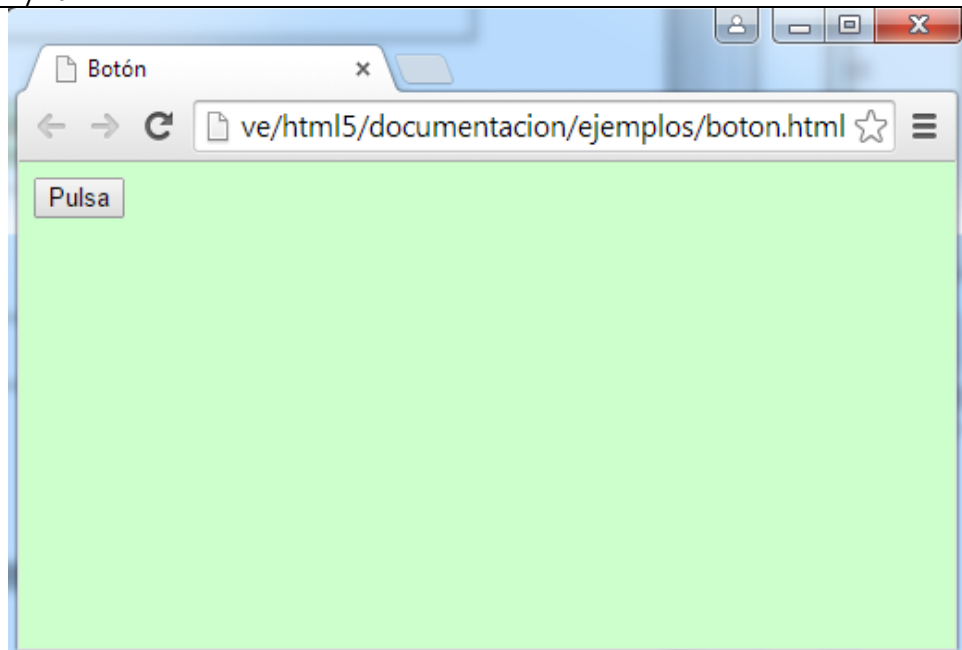


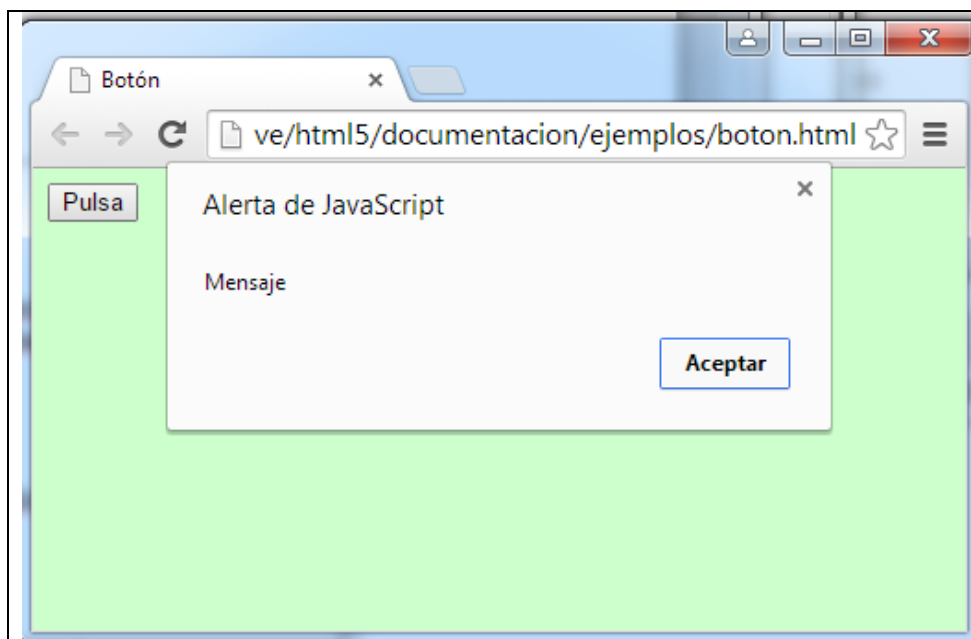
*Dentro de un elemento `<button>` se puede poner texto o imágenes a diferencia de los botones creados con `<input>` que no lo permiten.*

### Ejemplo:

```
<!DOCTYPE html>
<html lang="es-es">
 <head>
 <title>Botón</title>
 <meta charset="UTF-8">
 <style type="text/css">
 .fondoverde
 {background-color:#ccffcc;
 }
 </style>
 </head>
 <body class="fondoverde">
 <button type="button" onclick="alert('Mensaje')">Pulsa
 </button>

</body>
</html>
```





En este ejemplo hemos creado un botón que, cuando se pulsa, saca un mensaje con el texto “Mensaje”. Como se ve el botón responde a lo que se denomina un “evento” que no es ni más ni menos que una acción que, en este caso, ha sido generada por la pulsación del botón con el ratón. Al producirse dicho “evento” abre la ventana con el mensaje.

#### - <datalist>

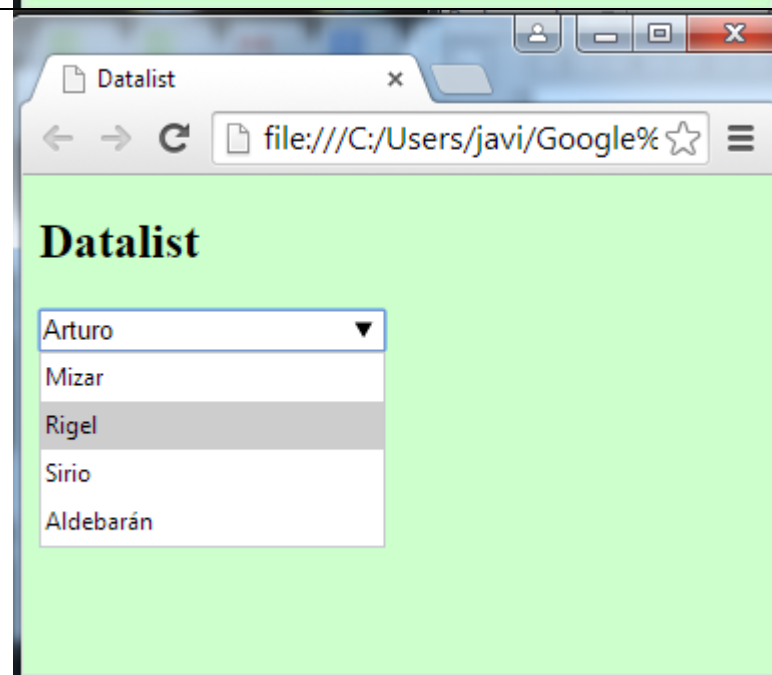
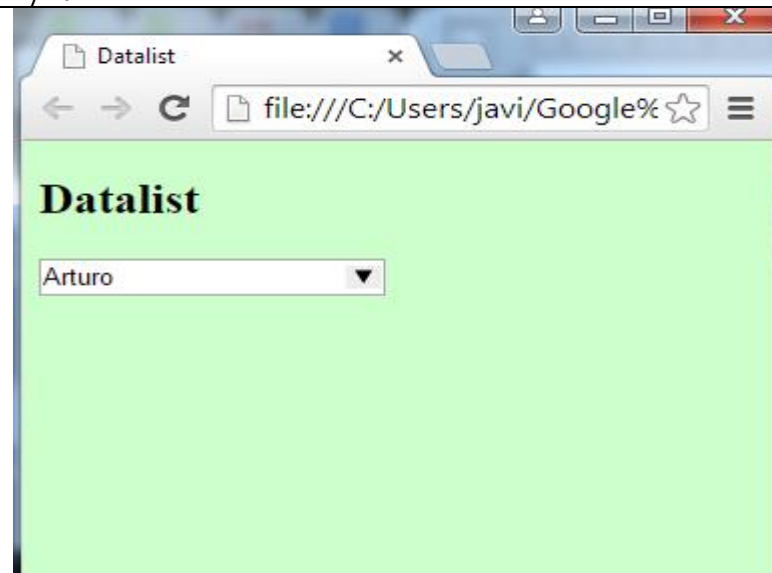
Especifica una lista desplegable de opciones predefinidas mediante el elemento <option> para un elemento <input> con atributo list.

Etiqueta	<datalist>					
Descripción	Especifica una lista de opciones predefinidas					
Elemento	En línea					
Navegadores que la soportan						
		20.0	4.0	10.0	9.0	Safari
Atributos	Globales y de eventos					
Atributos propios	Ninguno					
Diferencia entre html 4.01 y html5	Elemento nuevo HTML5					

#### Ejemplo:

```
<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>Datalist</title>
 <meta charset="UTF-8">
```

```
<style type="text/css">
 .fondoverde
 {background-color:#ccffcc; }
</style>
</head>
<body class="fondoverde">
<h2> Datalist</h2>
<form>
 <input list="estrellas" value="Arturo">
 <datalist id="estrellas">
 <option value="Mizar">
 <option value="Rigel">
 <option value="Sirio">
 <option value="Aldebarán">
 </datalist>
</form>
</body>
</html>
```



En este ejemplo se ve una lista desplegable que, cuando se pasa con el ratón por encima el cuadro de diálogo, aparece un icono ▼ que permite desplegar la lista de valores posibles de un elemento `<input>` (en este caso son nombres de estrellas).

	<i>El atributo <b>id</b> del elemento <code>&lt;datalist&gt;</code> debe coincidir con el valor del atributo <b>list</b> del elemento <code>&lt;input&gt;</code></i>
-----------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------

#### - `<keygen>`

Genera una clave privada, que es cargada localmente, y otra pública que es enviada a un servidor.

Se emplea siempre dentro de un formulario para el envío seguro de datos.

Etiqueta	<code>&lt;keygen&gt;</code>					
Descripción	Genera un par de claves para el envío seguro de datos					
Elemento	En línea					
Navegadores que la soportan						
	10	4.0	IE Explorer	11.0	5.1	
Atributos	Globales y de eventos					
Atributos propios	autofocus, challenge, disabled, form, keytype, name					
Diferencia entre html 4.01 y html5	Elemento nuevo HTML5					

#### Ejemplo:

```
<!DOCTYPE html>
<html lang="es-es">
 <head>
 <title>Keygen</title>
 <meta charset="UTF-8">
 <style type="text/css">
.fondoverde
{background-color:#ccffcc; }
 </style>
 </head>
 <body class="fondoverde">
 <h2> Keygen</h2>
 <form action="validacion.php">
Nombre:
<input type="text" name="user">

Encriptación: <keygen name="seguro">

<input type="submit">
 </form>
 </body>
</html>
```



The screenshot shows a web browser window with a single tab titled 'Keygen'. The address bar displays a file path: `file:///C:/Users/javi/Google%20Drive/html5/dk`. The page content is on a light green background and features the title 'Keygen' in a large, bold, black serif font. Below the title is a form with two fields: 'Nombre:' followed by a text input box containing the text 'Sandra', and 'Encriptación:' followed by a dropdown menu currently showing '2048 (Grado elevado)'. At the bottom of the form is a button labeled 'Enviar'.

En este ejemplo se enviaría en la variable “user” del formulario el valor de “Sandra” con la clave de encriptación en el variable “seguro”. Como se ve la encriptación está en una lista desplegable con dos posibles valores: 2048 (Grado elevado) o 1024 (Grado medio). Estas variables serían recogidas por el programa “validacion.php”.



## 5.6. EJEMPLO DE CONSTRUCCIÓN DE UN FORMULARIO CON ESTILOS

Los formularios no dejan de ser conjuntos de elementos HTML5 cuya información, introducida por el usuario, se envía a otros programas para ser procesada, por lo que la aplicación de estilos es prácticamente igual a que hemos visto en el tema de CSS, es decir, se crea la página html con el formulario (se deciden los tipos de elementos que forman parte del formulario y se les van aplicando estilos.

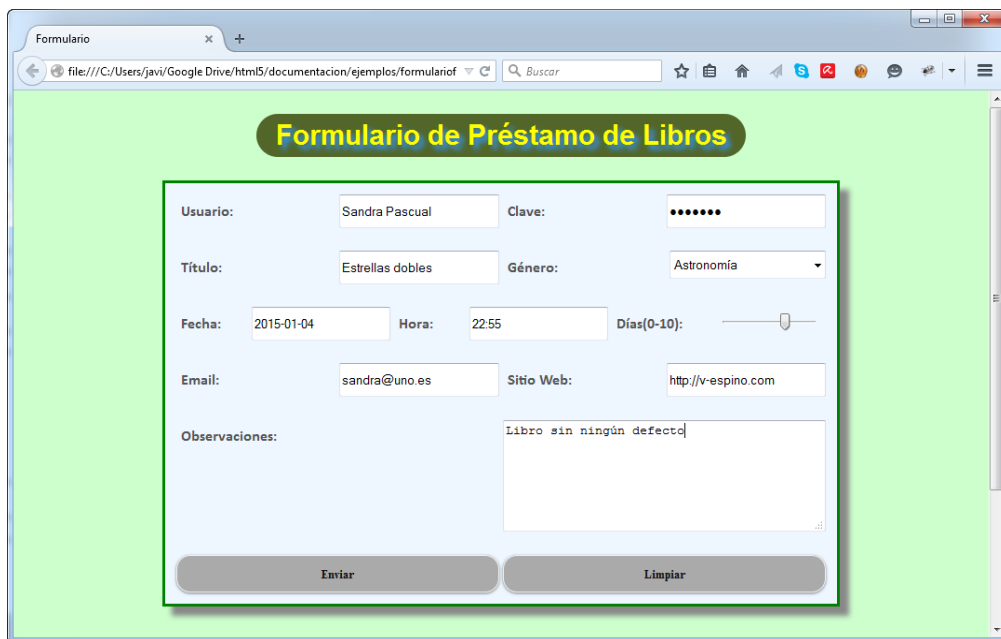
Un posible ejemplo de formulario podría ser el siguiente:

Queremos gestionar los préstamos de libros de una biblioteca pública a través de una página HTML5. Como necesitamos recolectar datos, aunque, en principio, no se envían a ningún sitio, vamos a incluir un formulario en nuestra página con los siguientes campos:

- *Usuario*: Nombre de la persona a la que se presta el libro.
- *Clave*: Clave de acceso que tiene la persona anterior.
- *Título*: Título del libro que se presta.
- *Género*: Temática del libro anterior que solo puede ser: astronomía, literatura, matemáticas, historia o física.
- *Fecha*: Fecha de préstamo.
- *Hora*: Hora de préstamo.
- *Días*: Días del préstamo que tiene como rango de 1 a 10 con el valor 7 por defecto.
- *Email*: Dirección de correo electrónico de la persona a la que se presta el libro.
- *Sitio web*: Página web, si la tuviera, de la persona a la que se presta el libro.
- *Observaciones*: Consideraciones a tener en cuenta sobre el préstamo que será un área de texto de 40 columnas y 6 filas.
- *Enviar*: Botón para enviar el formulario.
- *Limpiar*: Botón para limpiar los datos del formulario.

También le añadiremos un título a la página y haremos que el formulario quede centrado en la página sin ocupar todo el ancho.

Una posible distribución sería esta:



Ahora tenemos que decidir qué tipos de elementos usamos dentro del formulario y cuáles requeridos u obligatorios (atributo "required"). Podrían ser estos:

CAMPO	ELEMENTO Y TIPO
Usuario	<input type="text" required>
Clave	<input type="password" required>
Título	<input type="text" required>>
Género	<select>
Fecha	<input type="date" required>
Hora	<input type="time">
Días	<input type="range" required>
Email	<input type="email" required>
Sitio Web	<input type="url">
Observaciones	<textarea>
Enviar	<button type="submit">
Limpiar	<button type="reset">

También vamos a utilizar listas desordenadas para organizar los grupos de elementos. Cada fila del formulario será un elemento <ul> compuesto de elementos <li> formados por

un elemento `<label>` y el correspondiente elemento del formulario. Fíjate en la imagen para hacerte una idea.

Teniendo en cuenta lo anterior el código html quedaría así:

```
<!DOCTYPE html>

<html lang="es">

<head>

 <meta charset="utf-8">

 <title>Formulario</title>

</head>

<body>

 <h2>Formulario de Préstamo de Libros</h2>

 <form>

 <label>Usuario:</label>

 <input type="text" required>


```

```
<label>Clave:</label>

<input type="password" required>

 <label>Título:</label>

 <input type="text" required>

 <label>Género:</label>

 <select>

 <option value="Astronomía" selected>Astronomía</option>

 <option value="Literatura">Literatura</option>

 <option value="Matemáticas">Matemáticas </option>

 <option value="Historia">Historia </option>

 <option value="Física">Física </option>

 </select>

 <label>Fecha:</label>

 <input type="date" required>

 <label>Hora:</label>

 <input type="time">
```

```


 <label>Días(0-10): </label>

 <input type="range" min="0" max="10" value="7" required>

 <label>Email:</label>

 <input type="email" required>

 <label>Sitio Web:</label>

 <input type="url">

 <label>Observaciones:</label>

 <textarea cols="40" rows="6"></textarea>

 <button type="submit">Enviar</button>

 <button type="reset">Limpiar</button>
```

```

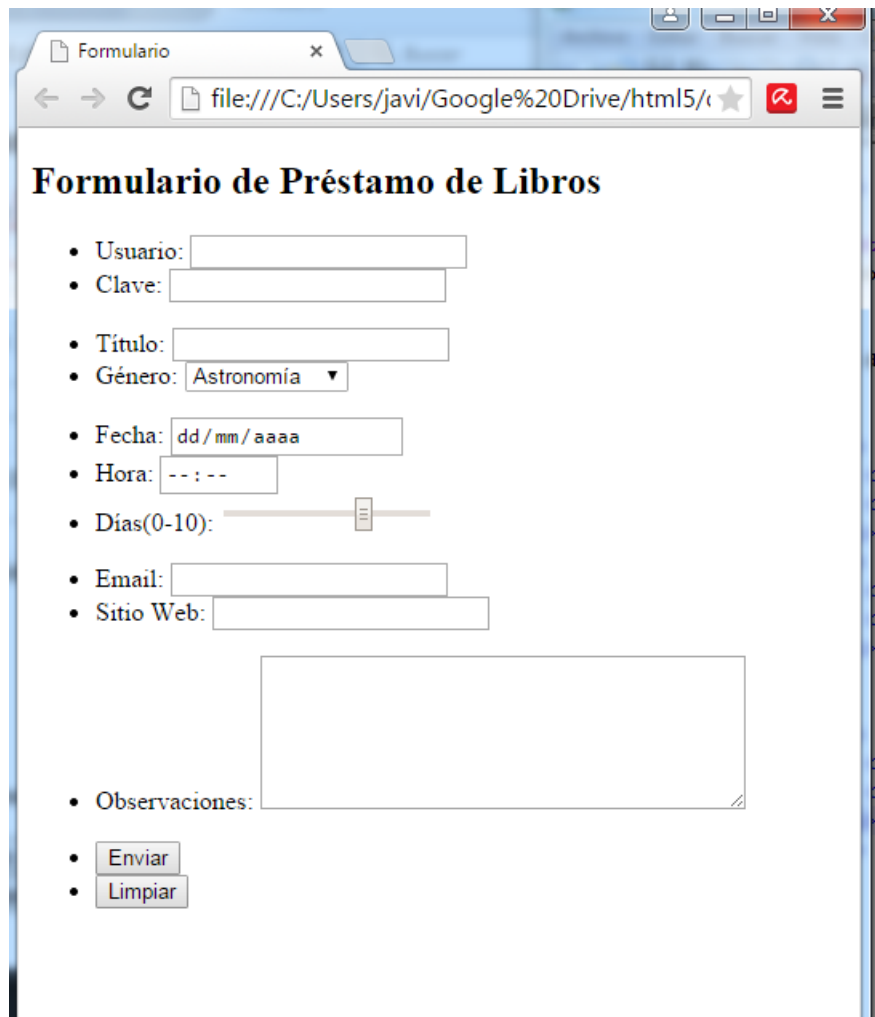

</form>

</body>

</html>

```

En el navegador se vería así:



A partir de aquí es cuando se empiezan a aplicar estilos

**Ejercicio 59:** Aplica estilos al ejemplo anterior utilizando contenedores y elementos flexibles para que, cuando cambies la resolución te quede de la siguiente manera:

Formulario

file:///C:/Users/javi/Google Drive

Buscar

### Formulario de Préstamo de Libros

Usuario:

Clave:

Título:

Género:

Fecha:

Hora:

Días(0-10):

Email:

Sitio Web:

Observaciones:

Sugerencia: El elemento `<form>` y los elementos `<ul>` son contenedores flexibles y el tamaño de la letra es más pequeño al cambiar la resolución.

**Ejercicio 60:** Para que veas como se envía la información a través de un formulario, vamos a enviar la información a la página `parámetros.html` que, a través de script javascript, devuelve los valores del formulario y vamos a visualizar dicha página en un

marco flotante, <iframe>, que pondremos justo debajo del formulario. Así, cuando se pulse el botón de “Enviar”, la información se ve en el marco.

Usa la siguiente declaración del elemento <form>

```
<form id="formulario" class="centrar" action="parámetros.html" method="get"
target="marco">
```

-----

```
</form>
```

Y esta para el elemento <iframe>

```
<iframe class="centrar" name="marco">
```

.....

```
</iframe>
```



## **BLOQUE 6:**

### **MULTIMEDIA**

## 6. MULTIMEDIA

### 6.1. INTRODUCCIÓN

El término multimedia hace referencia a una tecnología que permite el uso de gráficos, sonidos, etc. dentro de un entorno determinado donde elementos físicos electrónicos digitales almacenan y manejan los objetos anteriores y representan su información de una manera sugestiva y atrayente.



Una de las principales facetas de HTML5 es su fácil manejo de cualquier elemento multimedia ya que incorpora elementos específicos para usar estos objetos sin necesidad de instalar programas auxiliares (plug-in) en los navegadores siempre y cuando, éstos, soporten HTML5 lo que permiten la plena integración con otras tecnologías del navegador como JavaScript y CSS.

	<i>Los elementos multimedia transmiten dinamismo y calidad a tus páginas. No dudes en usarlos.</i>
------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------

	<i>Usa los elementos multimedia con sentido común ya que si abusas de ellos, lo normal es que oculten el auténtico contenido de tu página.</i>
-------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

## 6.2. VÍDEO

Un video digital es una secuencia de imágenes que, ejecutadas en secuencia, simulan movimiento. Se almacenan en un determinado formato digital de video como ser [AVI](#), [MPG](#), [RealVideo](#), [WMV](#), etc.

### 6.2.1. Formatos de vídeo

Cuando hablamos de formato de vídeo nos referimos a los tipos de archivos que contienen (contenedores), en la mayoría de los casos, tanto datos de audio como de vídeo así como la forma de sincronizar ambos datos para que la reproducción del archivo sea coherente (no puede ir más deprisa el sonido que la imagen o al revés).

Normalmente la información que lleva va comprimida y codificada con arreglo a unas determinadas reglas que forman lo que se denomina el códec de vídeo.



Un códec es un conjunto de normas y requisitos que permiten codificar la información y/o recuperarla de la forma más rápida y fiable posible.

Ejemplos de formatos de vídeo o contenedores son: .mpeg, .avi, .mov, .webm, .ogv, .mp4. Algunos de estos formatos están limitados a contener streams que se reducen a un pequeño juego de códec, mientras que otros son usados para objetivos más generales.

La necesidad de un formato de codificación base o códec base llevó al organismo responsable (W3C) a crear la especificación HTML5 a especificar que el formato de los vídeos debería ir en Theora, un codec de vídeo libre y que no tiene patentes, pero algunas empresas que componen el W3C se quejaron fuertemente (en especial Apple) ya que tenían **intereses comerciales** en usar sus propios codecs, y al final no se especificó un codec común.

Aunque, en teoría, cualquier códec puede ser encapsulado en un contenedor, tan solo se encuentran determinados códec en contenedores específicos, WebM, por ejemplo, ha sido definido para albergar únicamente VP8 y Vorbis. Ogg, suele acoger Theora, Vorbis, Speex o FLAC. MP4 normalmente contiene MP3, AAC y H.264.

La liberación del códec de vídeo VP8, sus mejores prestaciones frente a H.264, y su posterior apoyo por parte de los máximos responsables de los principales navegadores como son Opera, Chrome, Safari e, incluso, IEExplorer hacen que el formato de vídeo del futuro más inmediato sea WebM, que contendrá el vídeo codificado con VP8 y el audio con Vorbis.



El códec VP9 ha surgido hace poco como una evolución de VP8 y será su sucesor en el futuro.

### Tabla de formatos de vídeo:

Formato	Extensión	Descripción
AVI	.avi	AVI (Audio Video Interleave) fue desarrollado por Microsoft. El formato AVI es reconocido y manejado por la mayoría de los sistemas operativos y por casi todos los navegadores y muy usado en Internet.
WMV	.wmv	WMV ( <u>Windows Media Video</u> ) también ha sido desarrollado por Microsoft. El formato WMV es un formato común en Internet que los sistemas no son de Microsoft necesitan complementos especiales para manejarlos.
MPEG	.mpg .mpeg	El formato MPEG (Moving Pictures Expert Group) es el más popular en Internet. Es multiplataforma y lo manejan la mayoría de los navegadores.
<u>QuickTime</u>	.mov	QuickTime fue desarrollado por Apple. QuickTime es un formato común en Internet, pero los archivos QuickTime no pueden ser reproducidos bajo sistemas Windows sin un complemento específico que reconozca y maneje los mismos.
RealVideo	.rm .ram	RealVideo fue desarrollado por Real Media. RealVideo permite streaming de vídeo (vídeo en tiempo real, Internet TV). Los archivos comprimidos con RealVideo están libres de pérdida de datos. La calidad de los archivos de RealVideo son, por lo general, de calidad en términos comparativos, sin embargo, versiones más antiguas son, en comparación con formatos como por ejemplo MPEG, de menor calidad.
Flash	.swf .flv	Flash fue desarrollado por Macromedia. Flash requiere un componente extra para ser reproducido. Este componente viene preinstalado en la mayoría de los navegadores.
MP4	.mp4	Formato de archivo contenedor que forma parte del estándar MPG-4 parte 14 que permite streaming a través de la red. Utiliza el códec H.264
ogv	.ogg .ogv	Formato usado preferentemente para contenido audiovisual abierto, o sea, sin derechos de autor que utiliza los códec Theora, Vorbis o FLAC entre otros.

#### 6.2.2. Elemento <video>

Sirve para mostrar el contenido audiovisual (vídeo o animación) sin necesidad de nada más, es decir, sin tener que usar Flash o instalar codecs adicionales.

Etiqueta	<video>				
Descripción	Elemento para la reproducción de vídeo				
Elemento	Bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	4.0	3.5	9.0	10.5	4.0
Atributos	Globales y eventos				
Atributos propios	autoplay, controls, height, loop, muted, poster, preload, src, width				
Diferencia entre html 4.01 y html5	El elemento <video> es nuevo en HTML5				

Atributos propios	Valor	Descripción
autoplay 	Sin valor	Indica si se hace o no la reproducción automática del vídeo cuando se carga la página. Es un valor booleano, es decir, que toma valor verdadero o falso, y que si no se pone en la etiqueta vídeo toma el valor falso (no se reproduce) y si se pone sin valores toma el valor verdadero y reproduce automática el vídeo.
controls 	Sin valor	Es un booleano que indica al navegador que incorpore controles de usuario. El tamaño y formato de estos controles difiere de un navegador a otro.
height 	<i>pixels</i>	Especifica la altura del elemento
loop	Sin valor	Indica la repetición automática del vídeo cuando éste acabe. También es un valor booleano.

		
muted 	Sin valor	Especifica si el vídeo se reproducirá sin sonido
name 	Texto	Especifica el nombre del elemento
poster 	URL	Se utiliza para visualizar una imagen mientras carga el vídeo.
preload 	none  auto  metadata	Se utiliza para almacenar temporalmente archivos de gran tamaño.
src 	URL	Indica la dirección URL del vídeo a reproducir.
width 	pixels	Indica la anchura del elemento.

El elemento `<video>` tiene etiqueta de apertura (`<video>`) y de cierre (`</video>`) y dentro de ellas puede haber otros elementos hijos como `<source>` o `<track>`. Cualquier sentencia incluida dentro de `<video>` que no esté dentro de alguno de sus hijos específicos es denominada *contenido de retroceso*, que se mostrará cuando un navegador Web no soporte tanto el elemento `<video>` como `<audio>`.

### Ejemplo:

```
<video src="video.mp4" width="320">
```

El navegador no soporta el elemento `<video>`;

```
</video>
```

En el ejemplo si el navegador soporta el elemento <video> intenta reproducir el archivo “video.mp4” en una caja de 320 pixels de ancho y, si no lo soporta, devuelve el mensaje “El navegador no soporta el elemento <video>”.

- **Formatos que soportan los navegadores con el elemento <video>**

Navegador	mp4	webm	ogv
Internet Explorer 9+	SI	NO	NO
Chrome 6+	SI	SI	SI
Firefox 3.6+	SI	SI	SI
Safari 5+	SI	NO	NO
Opera 10.6+	SI (Opera 25+)	SI	SI



**mp4** es el formato más utilizado en Internet y el que reconocen todos los navegadores por lo que procura utilizarlo siempre que puedas.

- **Tipos MIME asociados a los formatos de vídeo**

Formato	mp4
mp4	video/mp4
webM	Video/webm
ogv	Video/ogg



Los tipos **MIME** indican al navegador cómo tienen que tratar el formato del archivo en cuestión.



No hay un MIME para **avi** por lo que tendremos que utilizar el elemento <object> para reproducirlos.

### 6.2.3. El elemento <source>

Este elemento sirve para especificar recursos multimedia, en especial ficheros de audio y vídeo, para los elementos <video> y <audio> y que el navegador pueda escoger cuál reproducir de acuerdo a los formatos que soporte.

Etiqueta	<source>					
Descripción	Recurso que se reproduce en los elementos <video> y <audio>					
Elemento	Ni bloque ni en línea					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
	4.0	3.5	9.0	10.5	4.0	
Atributos	Globales y eventos					
Atributos propios	media, src y type					
Diferencia entre html 4.01 y html5	El elemento <source> es nuevo en HTML5					

Atributos propios	Valor	Descripción
media 	<i>dispositivo</i>	Especifica el tipo de dispositivo y como se va a reproducir el fichero en él.
src 	URL	Dirección URL donde se encuentra el archivo a reproducir.
media 	<i>MIME</i>	Especifica el tipo MIME del archivo a reproducir.



Para reproducir un vídeo tenemos dos posibilidades:

- Utilizar el atributo “src” de <video>: Esta opción sólo nos permite indicar un único archivo de vídeo lo que implica que debemos acertar con el formato adecuado para que sea reproducido por el máximo número de navegadores.
- Usar el elemento <source>: Nos permite indicar varios archivos de vídeo y que el navegador escoja cuál sea más compatible con él a costa de ocupar más espacio.

	<i>El empleo de varios &lt;source&gt; da la posibilidad de reproducir el vídeo en uno de los formatos que puede soportar cualquier navegador.</i>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------

	<i>El empleo de varios &lt;source&gt; nos obliga a tener un mismo archivo de vídeo en diferentes formatos lo que conlleva un mayor uso de espacio de almacenamiento.</i>
-----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Ejemplo:

```
<video width="320" height="240">
 <source src="video.mp4" type="video/mp4">
 <source src="video.ogv" type="video/ogg">
 El navegador no soporta el elemento <video>
</video>
```

En este ejemplo existen dos posibles casos de ejecución:

- *Que el navegador en el que se está visualizando la página soporte el elemento <video>:* En este caso intentaría reproducir el archivo con el formato más compatible a dicho navegador o no lo reproduciría sino reconoce ninguno de los dos.
- *Que el navegador en el que se está visualizando la página no soporte el elemento <video>:* En este caso devuelve el mensaje “El navegador no soporta la elemento <video>”.

#### 6.2.4. El elemento <track>

Permite agregar pistas de texto que se sincronizan con el contenido del vídeo. Se utiliza dentro del elemento <video>.

Etiqueta	<track>				
Descripción	Agregan pistas de texto a un vídeo				
Elemento	Ni en bloque ni en línea				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	18.0	31.0	10.0	15.0	6.0
Atributos	Globales y eventos				
Atributos propios	default, kind, label, src, srclang				
Diferencia entre html 4.01 y html5	El elemento <track> es nuevo en HTML5				

Atributos propios	Valor	Descripción
default     	Sin valor	Especifica que la pista será activada si el usuario no indica otra pista.
kind     	captions chapters descriptions metadata subtitles	Indica la naturaleza de la pista que se va a utilizar.
label     	Texto	Especifica el título de la pista.
src     	URL	Especifica la dirección del archivo que va a contener la pista.

<b>srclang</b> 	<i>Idioma</i>	Especifica el idioma en el que está escrito el texto de la pista. Es obligatorio especificarlo cuando kind="subtitles".
-----------------------------------------------------------------------------------------------------	---------------	-------------------------------------------------------------------------------------------------------------------------

	<i>Una pista de texto puede proporcionar una mejor interpretación del contenido de un vídeo además de dotarlo de mayor accesibilidad.</i>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------

## 6.3. AUDIO

El sistema de audio es el sistema de grabación, transmisión y reproducción del sonido. Al igual que los vídeos, los archivos de sonido necesitaban en los antiguos navegadores los plug-ins correspondientes para poder ser reproducidos. De la misma manera que elemento <video> nació para que los navegadores no necesitaran ningún software especial para reproducir los vídeos, el elemento <audio> pretende lo mismo pero con los archivos de sonido.

### 6.3.1. Formatos de audio

Un formato de archivo de audio es un fichero en cuyo interior guarda una grabación de audio (música, voz, etc.)

Al igual que para el vídeo, existen también códecs específicos para el audio de formato abierto (GSM, VOX, etc.), de formato abierto libre (Flac, Ogg, Vorbis, etc.) y formatos propietarios (mp3, Aac, wma, wav, etc.).

Formato	Extensión	Descripción
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface) es un formato de dispositivos musicales electrónicos como sintetizadores y tarjetas de sonido. Se caracteriza por lo poco que ocupa sus archivos. MIDI es soportado por la mayoría de los navegadores.
MP3	.mp3	Los ficheros MP3 son actualmente un tipo de fichero MPEG. MPEG fue originalmente creado por Moving Pictures Experts Group. MP3 es el más popular formato en Internet de música. El sistema de codificación combina una buena compresión (archivos pequeños) con una alta calidad.
RealAudio	.rm .ram	RealAudio fue desarrollado por Real Media. Permite streaming de audio (música en tiempo real, Internet radio)
WAV	.wav	WAVE (más conocido por WAV) fue desarrollado por IBM and Microsoft. WAVs son compatibles con Windows, Macintosh y sistemas operativos Linux.

En la siguiente tabla están los códecs de audio compatibles de forma nativa con los diferentes navegadores

Navegador	MP3	Wav	Ogg Vorbis
Internet Explorer	SI	NO	NO

Chrome	SI	SI	SI
Firefox	SI	SI	SI
Safari	SI	SI	NO
Opera	SI	SI	SI

Y en esta tabla están los tipo MIME para los formatos de audio

Formato	MIME-type
MP3	audio/mpeg
Ogg	audio/ogg
Wav	audio/wav

### 6.3.2. Elemento <audio>

Se emplea para la reproducción de un archivo de audio.

Etiqueta	<audio>					
Descripción	Elemento para la reproducción de audio					
Elemento	Bloque					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
	4.0	3.5	9.0	10.5	4.0	
Atributos	Globales y eventos					
Atributos propios	autoplay, controls, loop, muted, preload, src					
Diferencia entre html 4.01 y html5	El elemento <audio> es nuevo en HTML5					

Atributos propios	Valor	Descripción
autoplay 	Sin valor	Indica si se hace o no la reproducción automática del fichero de audio cuando se carga la página. Es un valor booleano, es decir, que toma valor verdadero o falso, y que si no se pone en la etiqueta audio toma el valor falso (no se reproduce) y si se pone sin valores toma el valor verdadero y reproduce automática del archivo de audio.
controls 	Sin valor	Es un booleano que indica al navegador que incorpore controles de usuario. El tamaño y formato de estos controles difiere de un navegador a otro.
loop 	Sin valor	Indica la repetición automática del fichero de audio cuando este acabe. También es un valor booleano.
muted 	Sin valor	Especifica si el archivo de audio se reproducirá sin sonido
name 	Texto	Especifica el nombre del elemento
preload 	none  auto  metadata	Se utiliza para almacenar temporalmente archivos de gran tamaño.
src 	URL	Indica la dirección URL del fichero de audio a reproducir

## 6.4. ELEMENTOS <object> Y <embed>

Los elementos <object> y <embed> permiten insertar elementos multimedia (sonidos, vídeos, imágenes, etc.) en una página html5.

Si queremos emplear flash, visualizar un archivo .avi, o cualquier otro tipo de formato que no se reconoce con la etiqueta <audio> o <video> se puede intentar reproducir con estos dos elementos.



Tanto <object> como <embed> a través de la información que ofrece el tipo MIME indicado en el parámetro type de ambos elementos permiten reproducir el fichero en cuestión.

- Plug-in

Es una aplicación auxiliar que utiliza otra aplicación para que esta última tenga mayores prestaciones o, en definitiva, le permita realizar una determinada función. Un ejemplo es el plug in de Adobe Flash Player para Chrome que hace que este navegador pueda reproducir elementos flash.



Las etiquetas <embed> y <object> nos permiten insertar plug-ins en nuestra página html.



La filosofía de html5 es no utilizar estos programas auxiliares pero cuando un navegador tiene que abrir un fichero .avi, por ejemplo, el cual no puede ser reproducido con el elemento <video>, no queda más remedio que recurrir a los plug-in para poder visualizarlo.

- Elemento <object>

Este elemento permite incrustar o insertar objetos tales como vídeos flash, ficheros de audio, applets java, etc., dentro de un documento html. La gran mayoría de estos objetos necesitan un plug-in para que el navegador pueda reproducirlos.

Etiqueta	<object>					
Descripción	Inserta un objeto dentro de un documento html					
Elemento	Bloque y en línea					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					

Atributos propios	data, form, height, name, type, usemap
Diferencia entre html 4.01 y html5	- Bastantes atributos de HTML4 han sido suprimidos en HTML5    - form es atributo nuevo en HTML5    - En HTML5 se pueden enviar objetos a través de formularios    - En HTML5, &lt;object&gt; aparece en &lt;body&gt; en vez de en &lt;head&gt;

Atributos propios	Valor	Descripción
data 	URL	Dirección URL el objeto a reproducir
form 	Identificador de formulario	Especifica uno o más formularios a los que el objeto pertenece
height 	pixels	Especifica la altura del objeto
name 	Texto	Especifica el nombre del objeto
type 	Tipo MIME	Indica el tipo MIME asociado al objeto
usemap 	#mapa activo	Indica el mapa activo que se usará con el objeto
width 	pixels	Indica la anchura del objeto

Al igual que <video>, <object> permite definir alternativas dentro de él para que el navegador pueda escoger cual es la más compatible de todas, además de permitir



contenido de retroceso para los navegadores que no soporten el elemento `<object>`.

La principal diferencia entre `<video>` y `<object>` a la hora de indicar las alternativas es que `<video>` necesita el elemento `<source>` y con `<object>` se insertan otros `<object>` dentro él.

Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Objetos incrustados</title>

 <meta charset="utf-8">

</head>

<body>

 <object data="video.swf" type="application/x-shockwave-flash">

 <object data="video.mpeg" type="application/mpeg">

 <object data="video.gif" type="image/gif">

 El navegador no soporta el elemento <object>

 </object>

 </object>

 </object>

</body>

</html>
```

En este ejemplo el navegador tiene la posibilidad de reproducir un archivo flash (video.swf), otro archivo mpeg (video.mpeg) o visualizar una imagen (video.gif). De entre los tres escogerá el que le sea más compatible y lo reproducirá con el plug-in asociado que se indica en el parámetro type o, en el caso de que el navegador no reconozca el elemento `<object>`, se visualizará el mensaje "El navegador no soporta el elemento `<object>`".

Otro detalle importante es ver que los elementos `<object>` van dentro `<body>`.

	Lo más importante a la hora de usar el elemento <object> es indicar en el parámetro type el tipo MIME correcto para usar el plug-in adecuado
-----------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------

- Elemento <param>

Permite pasar parámetros adicionales de funcionamiento a los objetos definidos en el elemento <object> como autoplay para la reproducción automática. Se utiliza siempre con el elemento <object>

Etiqueta	<param>					
Descripción	Define un parámetro adicional de un objeto					
Elemento	Ni en bloque ni en línea					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales y eventos					
Atributos propios	name, value					
Diferencia entre html 4.01 y html5	Los atributos de HTML4 value y valuetype han sido suprimidos en HTML5					

Atributos propios	Valor	Descripción
name 	Texto	Especifica el nombre del parámetro
value 	Texto	Valor del parámetro

Los nombres de los parámetros (atributo name) no son fijos y dependen del tipo de objeto.

**Ejemplo:**

```

<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Objetos incrustados</title>

 <meta charset="utf-8">

</head>

<body>

 <object data="video.swf" type="application/x-shockwave-flash">

 <param name="autoplay" value="true"> </param>

 </object>

</body>

</html>

```

En este ejemplo el objeto definido en <object>, un vídeo flash, se le asigna el parámetro "autoplay" con el valor "true", lo que significa que el vídeo se reproducirá automáticamente cuando se cargue la página.

- Elemento <embed>  
Permite insertar un objeto en un documento html al igual que <object> pero no deja definir alternativas por lo que hay que acertar con el plug-in para que el archivo sea reproducido fidedignamente.

Etiqueta	<embed>					
Descripción	Inserta un objeto en un documento html					
Elemento	En bloque					
Navegadores que la soportan						
		Chrome	Firefox	IE Explorer	Ópera	Safari
Atributos	Globales y eventos					
Atributos	height, src, type, width					

propios	
Diferencia entre html 4.01 y html5	Los atributos de HTML4 value y valuetype han sido suprimidos en HTML5

Atributos propios	Valor	Descripción
height 	pixels	Especifica la altura del objeto
src 	URL	Especifica la dirección URL del objeto
type 	Tipo MIME	Indica el tipo MIME asociado al objeto
width 	pixels	Indica la anchura del objeto

### Ejemplo:

```

<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Objectos embed</title>

 <meta charset="utf-8">

</head>

<body>

 <embed src="video.swf" type="application/x-shockwave-flash"

 width="300" height="200">

</body>

</html>

```

En este ejemplo se inserta un vídeo flash con un ancho de 300 pixels y una altura de 200 pixels.

- Uso conjunto de <embed> y <object>

En el contenido de retroceso de <object> puede haber un elemento <embed> con el fin de que si el navegador no reproduce el archivo definido en <object> intente hacerlo con el que está definido en <embed>.

También <embed> tiene contenido de retroceso por lo que podría darse el caso contrario del primer párrafo, es decir, que <object> esté dentro de las etiquetas de apertura y cierre de <embed>

### Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Objetos embed</title>

 <meta charset="utf-8">

</head>

<body>

 <object type="application/x-shockwave-flash" data="video.swf">
 <embed src="video.swf" type="application/x-shockwave-flash"
 width="300" height="200">
 </object>

</body>

</html>
```

En este ejemplo si el navegador no reproduce el archivo "video.swf" con el elemento <object>, lo intenta con <embed>



Para reproducir archivos en formato avi o flash utilizaremos los elementos <object> y/o <embed> ya que no se puede hacer con el elemento <video>

**Ejemplo:**

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Objetos embed</title>

 <meta charset="utf-8">

</head>

<body>

 <object type="application/x-mplayer2"="video.avi">

 <embed src="video.swf" type="application/x-shockwave-flash"

 width="300" height="200">

 </object>

</body>

</html>
```

En este ejemplo el navegador intenta reproducir el archivo video.avi (formato avi) y, si no puede, intenta reproducir el archivo video.swf (formato flash).

- Uso conjunto de <video>, <object> y <embed>

Con la finalidad de que se visualice algún fichero independientemente del navegador que se utilice se puede utilizar los diferentes contenidos de retroceso de <video>, <object> y <embed>.

**Ejemplo:**

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Objetos embed</title>

 <meta charset="utf-8">

</head>

<body>
```

```
<video width="320" height="240">
 <source src="video.mp4" type="video/mp4">
 <source src="video.ogg" type="video/ogg">
 <object type="application/x-mplayer2" data="video.avi">
 <embed src="video.swf" type="application/x-shockwave-flash">
 </object>
 </video>
</body>
</html>
```

En este ejemplo el navegador abre una caja de 320 pixels de ancho por 240 de alto en el que intentará reproducir el archivo "video.mp4" o "video.ogg" dependiendo de cuál sea más compatible con él, si no puede visualizar cualquiera de los dos, intentará abrir el fichero "video.avi" y, en el caso de que no pueda hacerlo, intentará reproducir el archivo "video.swf"

## **BLOQUE 7: JAVASCRIPT**

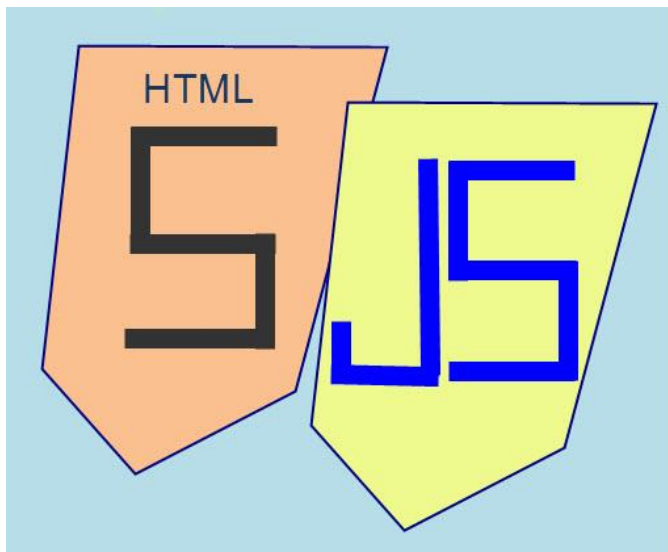


## 7. JAVASCRIPT.

### 7.1. GENERALIDADES.

Javascript es un lenguaje interpretado, es decir, un lenguaje que no necesita compilar los programas para poderlos ejecutar sino que el propio navegador se encarga de interpretarlos y ejecutarlos sin necesidad de pasos intermedios, que permite añadir mayores efectos dinámicos a los documentos html como, por ejemplo, botones que, al ser pulsados, realizan cambios en los elementos de la página web.

Dentro del Diseño Web, los lenguajes de scripting como Javascript van tomando una relevancia muy importante al utilizar una “programación del lado cliente”, es decir, que los elementos de esos lenguajes se tratan y manejan por los navegadores locales que, a su vez, consumen también recursos locales sin tener que depender de servidores remotos a no ser, por ejemplo, para cuestiones de seguridad como puede ser los procesos de identificación de usuarios.



Este auge de “la programación del lado cliente” ha hecho que el desarrollo de Javascript haya sido enfocado hacia una mayor portabilidad e integración con los modernos navegadores que también han aportado su granito de arena al aumentar su rendimiento e incorporar interfaces de programación de aplicaciones (APIs) como Web Storage, Canvas, etc., que asisten al lenguaje en funciones elementales con la idea de proporcionar poderosas funciones a través de técnicas de programación sencillas y estándares que facilitan la creación de programas útiles para el Diseño Web.



*Podemos ver HTML5 como la mezcla de tres elementos básicos: **HTML**, lenguaje para el contenido, **CSS**, herramienta para el aspecto, y **Javascript**, lenguaje para el dinamismo de los elementos.*

## 7.2. ELEMENTOS DE DEFINICIÓN DE SCRIPTS

- **<script>**

Permite la incorporación de programas de scripting de “lado cliente” como Javascript

Etiqueta	<script>					
Descripción	Permite la incorporación programas de scripting					
Elemento	Ni en bloque ni en línea					
Navegadores que la soportan						
	Chrome	Firefox	IE Explorer	Ópera	Safari	
Atributos	Globales					
Atributos propios	async, charset, defer, src, type					
Diferencia entre html 4.01 y html5	async es un nuevo atributo HTML5, xml:space ha sido suprimido y type es obligatorio en HTML4 pero opcional en HTML5					

Atributos propios	Valor	Descripción
async 	Sin valor	Permite cargar el contenido de la página html sin esperar a que acabe la ejecución del script. Solo vale para scripts externos
charset 	<i>Mapa de caracteres</i>	Mapa de caracteres empleado en un script externo.
defer 	Sin valor	El script se ejecuta cuando se acabe e cargar la página
src 	URL	Indica la dirección URL del script

type 	Tipo MIME	Especifica el tipo MIME del script. Sus posibles valores son:      • text/javascript (defecto)     • text/ecmascript     • application/ecmascript     • application/javascript

 → Nuevo en HTML5

Entre las etiquetas de apertura y cierre del elemento `<script>` van las instrucciones del programa y este elemento puede estar incluido dentro de `<head>` dentro de `<body>` dependiendo de cómo queremos la ejecución del mismo.

- **`<noscript>`**

Define un contenido alternativo para cuando el navegador tenga deshabilitada la ejecución de scripts o cuando dicho navegador no los soporte.

Etiqueta	<code>&lt;noscript&gt;</code>				
Descripción	Define un contenido alternativo cuando no se puede ejecutar un script				
Elemento	Ni en bloque ni en línea				
Navegadores que la soportan					
	Chrome	Firefox	Internet Explorer	Ópera	Safari
Atributos	Globales				
Atributos propios	No tiene				
Diferencia entre html 4.01 y html5	En HTML4 <code>&lt;noscript&gt;</code> solo puede usarse dentro de <code>&lt;body&gt;</code> y en HTML5 puede ir tanto en <code>&lt;head&gt;</code> como en <code>&lt;body&gt;</code>				

### Ejemplo:

```
<!DOCTYPE html>

<html lang="es-es">

<head>
```

```
<title>Script</title>

<meta charset="utf-8">

</head>

<body>

 <script type="text/javascript">

 document.write("Mensaje");

 </script>

 <noscript>

 O bien tu navegador no soporta scripts o los tiene deshabilitados

 </noscript>

</body>

</html>
```

En este ejemplo se ve que el elemento `<script>` del tipo “text/javascript”, o sea, javascript, está definido en `<body>` y se ejecutará al cargar la página siempre y cuando el navegador lo permita. Este pequeño programa contiene una única instrucción (`document.write(“Mensaje”;`) acabada en “;” que visualiza el mensaje “Mensaje” por pantalla cuando se ejecuta.

En el caso de no poder ejecutar el script visualizará el mensaje “O bien tu navegador no soporta scripts o los tiene deshabilitados”.

## 7.3. INCORPORACIÓN DE SCRIPTS A UN DOCUMENTO HTML

Los scripts se pueden incorporar de la misma forma que las hojas de estilo CSS, es decir, en línea, incorporados al propio documento o en archivos externos.

- **En línea**

Se incorpora el código dentro de la etiqueta de apertura de un elemento a través de un manejador de eventos como puede ser onclick (pulsar con el ratón sobre él), onMouseOver (que pase el puntero del ratón sobre él), onMouseOut (cuando salga el puntero del ratón de él) u otros más que se producen cuando se pulsa una tecla o se carga una página.

**Ejemplo:**

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Script en línea</title>

 <meta charset="utf-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 #uno {background-color:orange;}

 #dos

 { background-color:pink;}

 p

 {width: 300px;

 height: 150px;}

 </style>

</head>

<body class="fondoverde">
```

```
<h1> script en línea </h1>
```

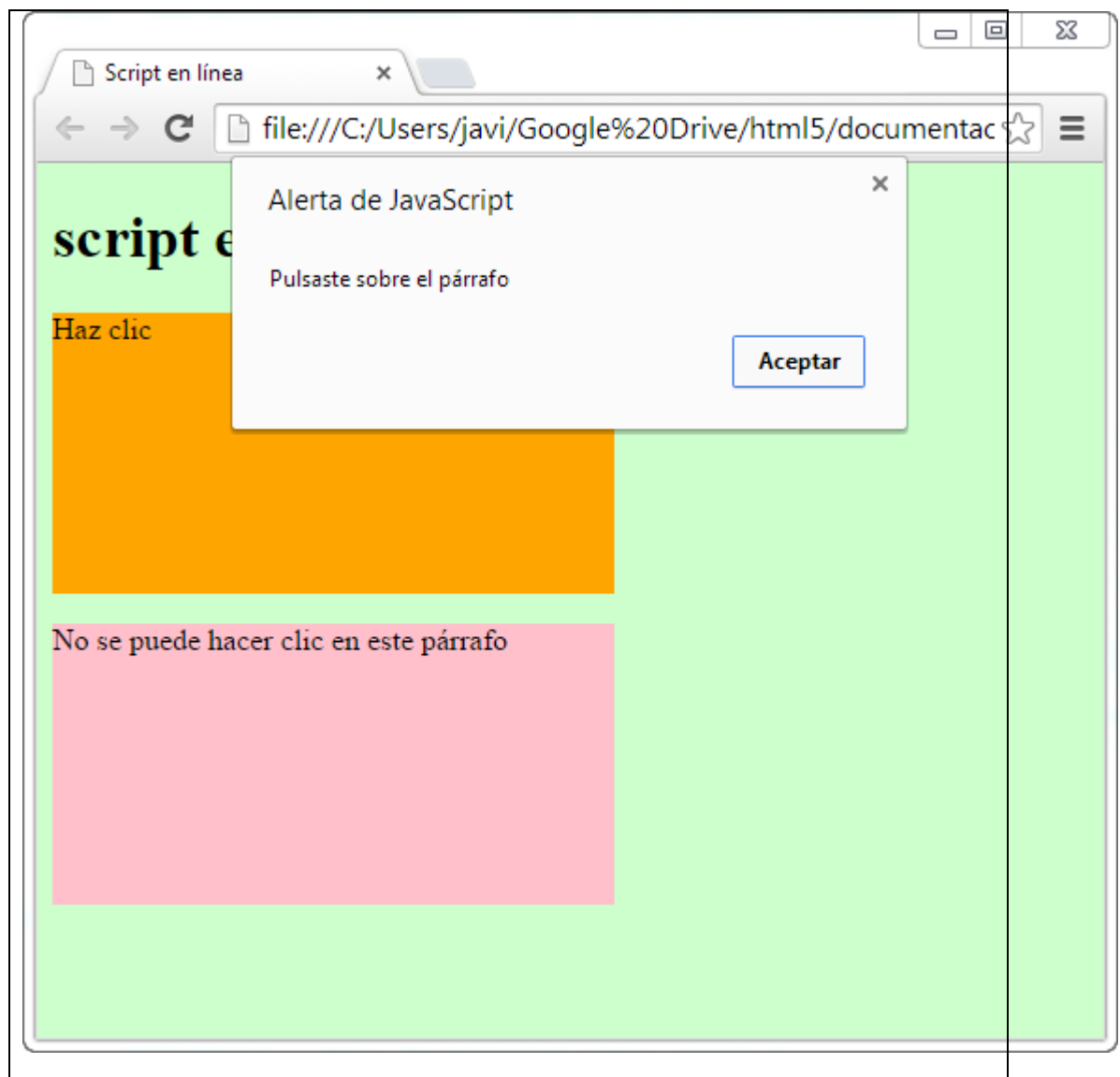
```
<p id="uno" onclick="alert('Pulsaste sobre el párrafo')">Haz clic </p>
```

```
<p id="dos"> No se puede hacer clic en este párrafo</p>
```

```
</body>
```

```
</html>
```





En este ejemplo en el elemento `<p>` con `id="uno"` y texto inicial `"Haz clic"` tiene definido que, cuando se haga clic sobre él con el ratón, saque una pequeña ventana con el texto "Pulsaste sobre el párrafo" (`onclick="alert('Pulsaste sobre el párrafo')"`)

- **Incrustado en `<head>` o `<body>`**

Es cuando el código del script se inserta entre las etiquetas de apertura y cierre del elemento `<script>` que puede estar en cualquier parte del documento html aunque es recomendable ponerlo dentro `<head>` por aislar lo que es html de lo que es javascript.

**Ejemplo:**

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Script en línea</title>

 <meta charset="utf-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 </style>

 <script type="text/javascript">

 alert("Hola Mundo");

 </script>

 </head>

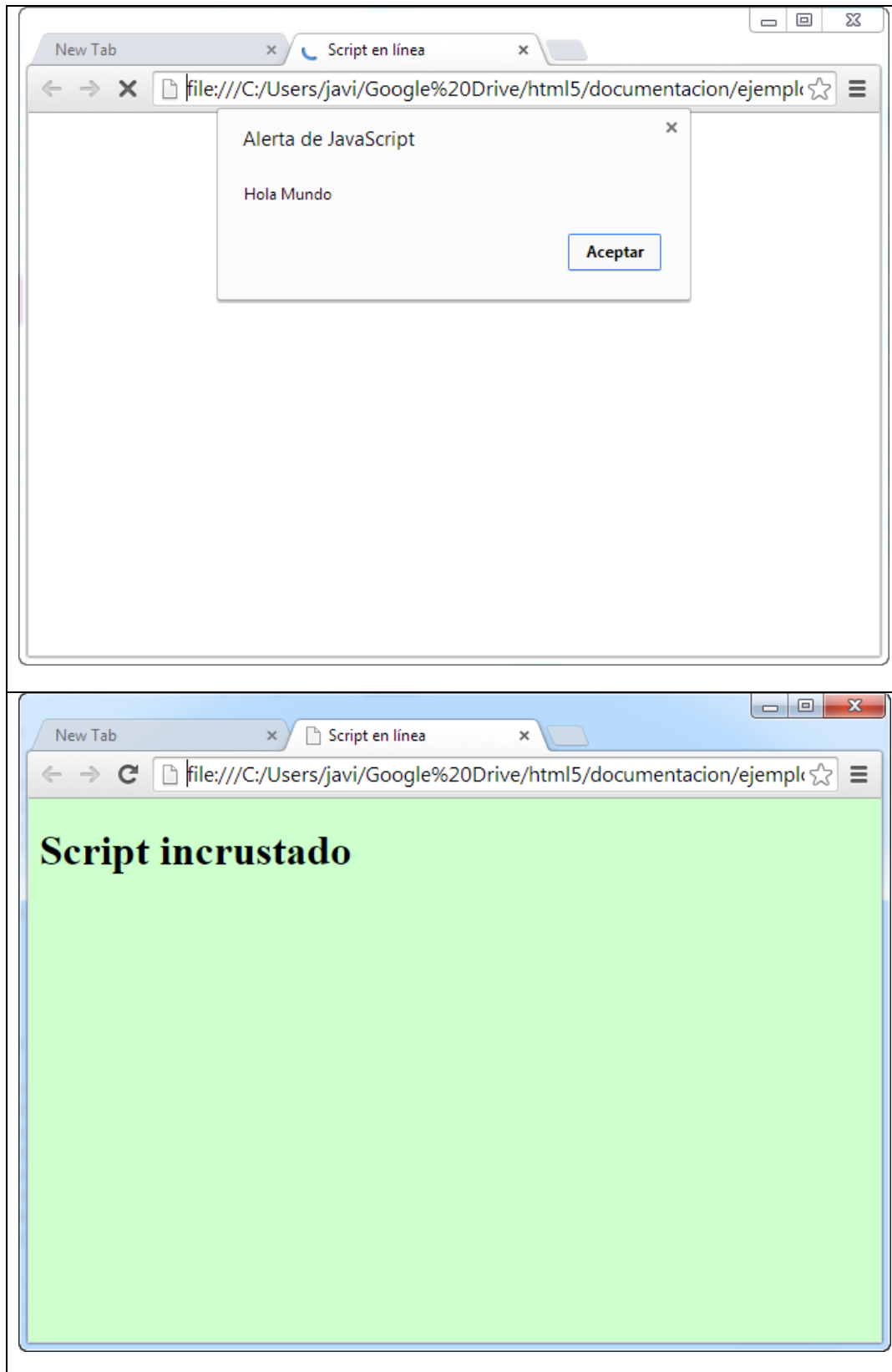
 <body class="fondoverde">

 <h1> Script incrustado</h1>

 </body>

</html>
```





En este ejemplo se define un script dentro `<head>` que abre una ventana de alerta de javascript con el mensaje “Hola Mundo”. Lo primero que haría el navegador es ejecutar el script y después de pulsar el botón de aceptar de la ventana de alerta se ejecutaría lo que hay en `<body>`.

	<i>Si pones código javascript dentro de &lt;head&gt; procura que sean funciones que se ejecuten cuando sucede algún evento ya que, si haces como en el ejemplo anterior, se ejecutará antes el código javascript que el contenido de la página web</i>
-----------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- **En un archivo externo**

Al igual que se hace con las hojas CSS externas, se puede guardar todo el código javascript en archivos de texto con extensión normalmente “.js” independientes de los documentos html lo que facilita su mantenimiento y su utilización en otras páginas web.

	<i>Podemos insertar tantos archivos externos javascript como queramos en nuestras páginas.</i>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------

**Ejemplo:**

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Script en línea</title>

 <meta charset="utf-8">

 <style type="text/css">

 .fondoverde

 {background-color:#ccffcc;}

 </style>

 <script type="text/javascript" src="archivo.js">

 </script>

</head>

<body class="fondoverde">

 <h1> Script incrustado</h1>

</body>
```

```
</html>
```

Contenido del archivo "archivo.js"

```
alert("Hola Mundo");
```

Este ejemplo realiza lo mismo que el anterior con la salvedad de que el código javascript está en el fichero "archivo.js" que se encuentra en la misma carpeta que la página html que lo utiliza.



*Si usas archivos externos puedes usar el parámetro **async** del elemento `<script>` lo que permite que se pueda cargar el contenido de la página sin tener que esperar a que acabe de ejecutarse el script*

## 7.4. PROGRAMACIÓN INICIAL

En este bloque vamos a ver los elementos y estructuras básicas de un programa javascript. Su sintaxis y lógica es muy parecida a otros lenguajes de programación como son Java o C pero existen sutiles diferencias que veremos a continuación.

### 7.4.1. Reglas sintácticas básicas

La sintaxis de un lenguaje de programación es el conjunto de normas que indican como se debe escribir un programa para que sea utilizado de forma correcta.

Las reglas básicas son las siguientes:

- No se indica el tipo de variable cuando se crea por lo que puede albergar diferentes tipos de datos durante la ejecución del script.
- Se distingue entre mayúsculas y minúsculas, es decir, si escribimos, por ejemplo, `aLert("Mensaje")` no se ejecutará nunca.
- Aunque la sintaxis de un script javascript es muy parecida a la de programas escritos en C o Java, una de las diferencias es que no es necesario acabar las instrucciones en ";" en javascript,
- Se ignoran los espacios en blanco repetidos y las líneas en blanco entre instrucciones.
- Se pueden incluir comentarios poniendo `//` delante de una línea de texto o poniendo `/*` Texto del comentario `*/`
- Al igual que cualquier lenguaje de programación, javascript contiene un conjunto de "palabras reservadas" que tienen un significado específico y se emplean para formar estructuras sintácticas concretas.



```

//="hugo"
//="25 mai 2011 19:14:28$"
function search(path,dir,i,taille): def search(path,dir,i,taille):
 action principale. Paramètres : chemin du fichier, dossier de travail, iteration n°.
 path.replace(dir,"") def search(path,dir,i,taille):
 g = name.replace("avi","").replace(" ", "").lower()
 url = "http://www.mipomik.fr/recherche/?q={0}".format(string)
 = urllib2.Request(the_url) string = name.replace("avi","").replace(" ", "").lstring
 .replace("avi","").replace(" ", "").lstring = name.rep
 dle = urllib2.urlopen(req)
 cept IOError, e: string = name.replace("avi","").replace(" ", "").l
 .avi","").replace(" ", "").l
 : hasattr(e, 'reason'): echo "musimgalerie.blogspot.com";
 print "Nous avons échoué à joindre le serveur."
 print "Raison: ", e.reason
 elif hasattr(e, 'code'):
 print "satisfaire la demande." string = name.replace("avi","").replace(" ", "").l
 print "Code d'erreur: ", e.code
 the read()

```

### 7.4.2. Variables

Una variable es un elemento que permite guardar datos de forma temporal y utilizar su valor para la realización de ciertas operaciones.

Existen variables de diferentes tipos dependiendo del valor que se les asignen como numéricas, cadenas de texto, boolean, etc. También se pueden definir colecciones de variables del mismo tipo dentro de *arrays* de una o varias dimensiones.

La palabra reservada "var" sirve para declarar una variable la primera vez que aparece en el programa y luego ya se referencia la variable por el nombre de la misma.

No es necesario inicializar una variable a la hora de declararla.

**Ejemplos de variables simples:**



```
var variable = "Esta es una cadena de texto";

var num1 = 4;

var num2 = 3;

var num3;

var suma = num1 + num2; /* num1 y num2 están declarados más arriba */

var logico = false; // logico es una variable boolean
```

### Ejemplos de arrays:

```
var impares = [1,3,5,7];

/* El array "impares" tiene 4 valores comenzando por el elemento impares[0], que
vale 1 y acabando en impares [3], que vale 7 */

var impares[4]=9;

/* Al array "impares" se le agrega un quinto valor */

var suma=impares[0]+impares[2];

/* A la variable "suma" se le asigna el valor 6 (1+5)*/

var pares[];

/* Se define el array "pares" para luego asignarle valores */

var tabla[][];

/* Se define el array "tabla" para luego asignarle valores */

tabla[1][3]="Elemento 1,3 de la tabla";

/*Apoyándonos en el ejemplo anterior asignamos al elemento
1,3 de la tabla la cadena de texto "Elemento 1,3 de la tabla" */
```

### 7.4.3. Operadores

Permiten manejar los valores de las variables y realizar cálculos tanto lógicos como aritméticos con ellos.



Tipo	Símbolo	Descripción	Ejemplo
Asignación	=	Carga un valor en una variable	num=3; <b>var</b> num1="hola";
Incremento (*)	++	Incrementa en una unidad una variable	<b>var</b> num=3; ++num; //num = 4
Decremento (*)	--	Decrementa en una unidad una variable	<b>var</b> num=3; --num; //num = 2
Negación	!	Devuelve el valor opuesto de la variable	<b>var</b> variable=true; alert(!variable); //devuelve false
AND	&&	Devuelve true si los dos valores son true o, en cualquier otro caso, falso	<b>var</b> varia1=true; <b>var</b> varia2=false; <b>var</b> a; a=varia1 && varia2; //a vale false
OR		Devuelve false si los dos valores son falsos o, en cualquier otro caso, verdadero	<b>var</b> varia1=true; <b>var</b> varia2=false; <b>var</b> a; a=varia1    varia2; //a vale true
Matemáticos básicos	+ - * /	Suman, restan, multiplican o dividen variables	<b>var</b> vari1=3 <b>var</b> vari2=2 <b>var</b> a=vari1- vari2; //a vale 1
Módulo	%	Calcula el resto de	<b>var</b> vari1=7;

		una división entera	<b>var</b> vari2=2; <b>var</b> a=vari1% vari2; //a vale 1
Mayor que	>	Devuelve true si la variable de la izquierda es mayor que la de derecha	<b>var</b> vari1=3 <b>var</b> vari2=2 <b>var</b> a=vari1>vari2; //a vale true
Menor que	<	Devuelve true si la variable de la derecha es mayor que la de izquierda	<b>var</b> vari1=3 <b>var</b> vari2=2 <b>var</b> a=vari1<vari2; //a vale false
Mayor o igual que	>=	Devuelve true si la variable de la izquierda es mayor o igual que la de derecha	<b>var</b> vari1=3 <b>var</b> vari2=3 <b>var</b> a=vari1>=vari2; //a vale true
Menor o igual que	<=	Devuelve true si la variable de la derecha es mayor o igual que la de izquierda	<b>var</b> vari1=3 <b>var</b> vari2=4 <b>var</b> a=vari1<=vari2; //a vale false
Igual que	==	Devuelve true si ambas variables son iguales	<b>var</b> vari1=3 <b>var</b> vari2=4 <b>var</b> a=vari1==vari2; //a vale false
Distinto de	!=	Devuelve true si ambas variables son distintas	<b>var</b> vari1=3 <b>var</b> vari2=4 <b>var</b> a=vari1!=vari2; //a vale true

(\*) Estos dos operadores pueden ir delante o detrás de la variable. En el caso de ir delante el incremento/decremento se hace antes de operar y en el caso de ir detrás se opera primero y se incrementa después.

#### 7.4.4. Estructuras de control

Permiten controlar e, incluso, alterar el flujo de ejecución de las instrucciones de cualquier programa.

- **if**

Si se cumple la condición definida en su construcción ejecuta las instrucciones definidas en su bloque

Formato	Ejemplo	Explicación
<b>if</b> (condición){ ..... }	<pre>var num1=1; var num2=2; If (num1&lt;num2) { var num3=num1+num2;   alert(num3); }</pre>	Se visualiza una ventana con el valor de num3 (suma de num1 y num2) ya que la condición se cumple

- **if .. else**

Si se cumple la condición definida en su construcción ejecuta el primer bloque de instrucciones y si no se cumple ejecuta el bloque de instrucciones definido detrás de la palabra reservada “else”.

Formato	Ejemplo	Explicación
<b>if</b> (condición){ ..... } <b>else</b> { .... }	<pre>var num1=1; var num2=2; If (num1&lt;num2) { mayor="no es";} else { mayor="es";} alert ("num1 " + mayor + " mayor que num2");</pre>	Si num2 es mayor que num1 asigna el valor “no es” a mayor y si no se cumple lo anterior asigna el valor “es” a mayor



- **for**

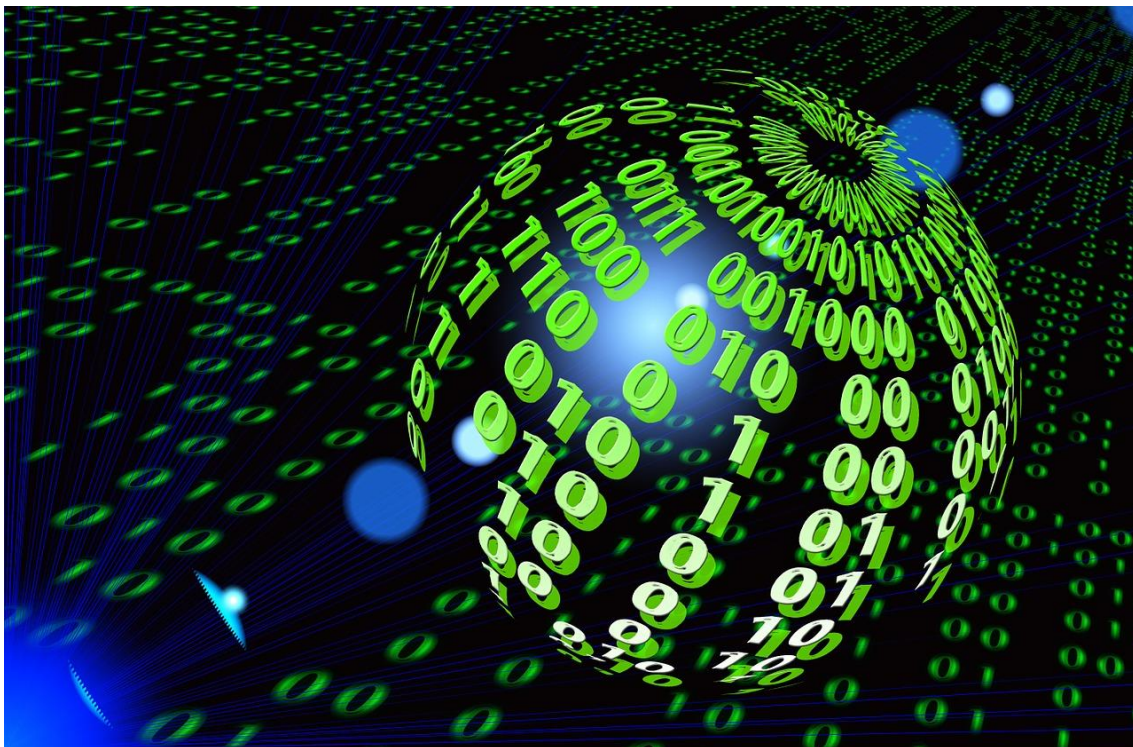
Estructura repetitiva que se ejecuta un número determinado de veces. Se compone de un valor inicial, una condición de final de bucle y un incremento/decremento.

Formato	Ejemplo	Explicación
<b>for</b> (ini;con;inc)  { ... }  /* Ini:Inicial Con; Condición Inc; incremento */	for (var n=0;n<5;n++)  { alert (n); }	Se define la variable n con valor 0 y se visualiza cinco veces una ventana con los números 0, 1, 2, 3 y 4 cada vez ya que n toma valores de 0 a 4

- **for .. in**

Variante de la estructura **for** vista anteriormente que permite recorrer los componentes de un array u objeto sin tener que conocer su número de componentes.

Formato	Ejemplo	Explicación
<b>for</b> ( n in array)  { ... }  /* n: Índice */	var vector = [1,2,3,4]  for (indice in vector) { alert (vector[indice]); }	Se define el array vector con cuatro componentes y luego se visualizan cuatro ventanas con los valores 1, 2, 3 y 4 que se corresponde con cada uno de los componentes del vector



- **while**

Estructura repetitiva que se puede cumplir 0 o más veces dependiendo de la condición definida en su construcción.

Formato	Ejemplo	Explicación
<b>while</b> (condición) { ... }	var n=1; while (n<7) { alert(n); n++; }	Visualiza seis ventanas (valores de 1 a 6 de n) mientras n sea menor que 7. Dentro del bucle se incrementa n para que llegue a 7 y la condición sea falsa

- **do .. while**

Estructura repetitiva similar a **while** que se ejecuta una o varias veces dependiendo de la condición definida en su construcción

Formato	Ejemplo	Explicación
<b>do</b> {	var n=1; do	Visualiza una única ventana con el valor 1 ya que la condición es falsa (2 no es mayor que 7) además

...	{	de incrementar en uno n.
}	alert(n);	
<b>while</b> (condición);	n++;	
	} while (n>7);	

- **switch**

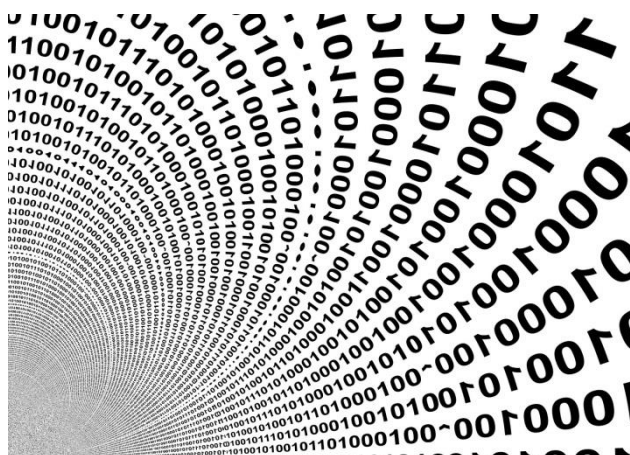
Estructura que sirve para comprobaciones múltiples que se puede usar en vez de una estructura if ..then .. else ..else.. demasiado compleja. La palabra reservada *case* indica cada uno de los posibles valores a evaluar.

Formato	Ejemplo	Explicación
<b>switch</b> (variable)  { <b>case</b> valor1: ... <b>break</b> ;  <b>case</b> valor2: ... <b>break</b> ;  <b>default</b> : ... <b>break</b> ; } 	var n=1; var mensaje; switch (n) { case 1: mensaje="hola"; break; case 2: mensaje="adiós"; break; default: mensaje="uno"; break; } alert(mensaje);	Si el valor de n es 1 se asigna el valor "hola" a mensaje, si el valor es 2 se asigna "adiós" y si no es ni 1 ni 2 se asigna el valor "uno" por lo que en este caso se visualizaría una ventana con el mensaje "hola" ya que n vale 1. La instrucción <i>break</i> sirve para romper la secuencia y no ejecutar las instrucciones que hubiera por debajo.

## 7.4.5. Funciones

Una función en cualquier lenguaje de programación es un bloque de instrucciones agrupado que realiza una operación concreta y que se puede utilizar tantas veces como se quiera a través de una llamada. La definición de la función lleva la palabra reservada **function** por delante seguida por paréntesis donde pueden ir parámetros pasados a dicha función.

Si a una variable se le asigna el valor que devuelve una función, esta función devolverá dicho valor con la palabra reservada **return**.



- **Formato del cuerpo de la función:**

```
function nombrefunción (parámetro1, parámetro2, ...)
{
 Instrucción1;

}

/* Una función puede no tener parámetros */
```

- **Llamada a la función**

```
Nombrefunción (parámetro1, parámetro2, ..);

var a = Nombrefunción (parámetro1, parámetro2, ...)

/* Si no lleva parámetros la función */

Nombrefunción ();

var a = Nombrefunción ();
```

- **Ejemplo:**

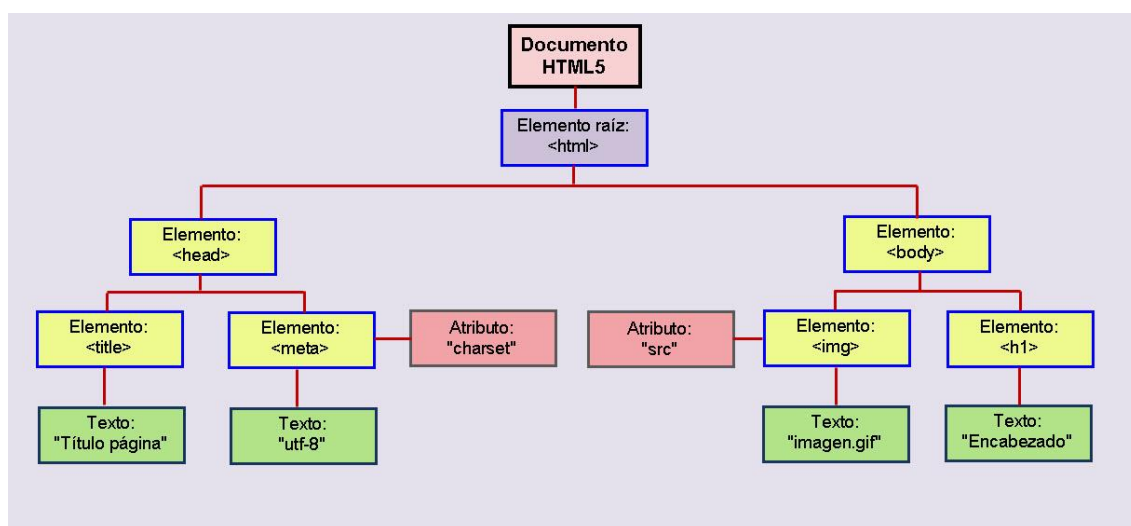
<pre><b>Function</b> sumar (num1, num2) {     var suma=num1+num2;     <b>return</b> suma; }  var n=1; var a=2; var sum=sumar(n,a); alert(sum);  var j=1; var m=4; sum=sumar(j,m) alert(sum);</pre>	La función <i>sumar</i> suma dos números pasados como parámetros guardando dicha suma en la variable <i>suma</i> y devolviendo dicho valor con la palabra reservada <b>return</b>.   num1, num2 y suma son variables locales a la función y solo tienen validez en ella. En cambio n, a, j y m son variables globales que valen para todo el script.   En la primera ventana se visualiza 3 (num1 se corresponde con n y num2 con a) y en la segunda ventana se visualiza 5 (num1 es j y num2 m)
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 7.5. DOM

El **Document Object Model** o **DOM** (Modelo de Objetos del Documento) es una de las innovaciones en la construcción de páginas web dinámicas que permite el acceso y la manipulación de cualquier componente de un documento html ( elementos, atributos, estilos, etc.) a través de un conjunto estándar de objetos que forman una estructura jerárquica de nodos.

El consorcio W3C se encarga de mantener las especificaciones del DOM y lo define de la siguiente manera:

*El Modelo de Objetos del W3C documento (DOM) es una plataforma y una interfaz que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de un documento*



Prácticamente todos los lenguajes actuales tienen estructuras y elementos que les permiten realizar programación orientada a objetos. Javascript no es una excepción y, aunque no dispone de todas las características de los lenguajes orientados a objetos (ni la herencia ni el polimorfismo), permite la creación, definición de métodos y atributos de objetos a través de lo cual puede manipular sin problemas la estructura que crea el DOM.

	<i>Un <b>objeto</b> es una entidad que dispone de propiedades o atributos y de operaciones específicas o métodos</i>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

En el DOM, todos los elementos HTML están definidos como objetos, una propiedad es un valor que se puede obtener o usar como, por ejemplo, cambiar el contenido de un elemento HTML, y, por último, un método es una acción que se puede hacer sobre el objeto como, por ejemplo, añadir o eliminar un elemento HTML.

En el DOM HTML, el objeto *document* representa una página web y todos los demás objetos de dicha página descienden directa o indirectamente de él, es decir, es el nodo raíz.

- **Métodos DOM**

A través de los métodos siguientes se puede buscar, acceder, crear, borrar o cambiar tanto elementos como sus valores asociados como pueden ser sus atributos o propiedades de estilo

Método	Descripción
<code>document.getElementById()</code>	Búsqueda de elementos por <b>id</b>
<code>document.getElementsByTagName()</code>	Búsqueda de elementos por nombre de etiqueta
<code>document.getElementsByClassName()</code>	Búsqueda de elementos por nombre de clase CSS
<code>Document.getElementsByName()</code>	Búsqueda de elementos por nombre de elemento
<code>Elemento.innerHTML="texto"</code>	Cambia el contenido de un elemento
<code>Elemento.atributo="valor"</code>	Cambia el valor del atributo de un elemento
<code>Elemento.setAttribute(atributo, valor)</code>	Cambia el valor del atributo de un elemento
<code>Elemento.style.propiedad="valor"</code>	Cambia el una propiedad de estilo de un elemento
<code>document.createElement("elemento")</code>	Crea un elemento HTML
<code>document.removeChild("elemento")</code>	Borra un elemento HTML
<code>document.appendChild("elemento")</code>	Añade un elemento HTML
<code>document.replaceChild("elemento")</code>	Reemplaza un elemento HTML
<code>document.write("texto")</code>	Escribe "texto" como contenido de la página web

- **Propiedades DOM**

Con las siguientes propiedades podemos acceder a los elementos en general de un documento html.

Propiedad	Descripción
<code>document.anchors</code>	Devuelve todos los elementos <a> que tengan definido el atributo <i>name</i>



document.baseURI	Devuelve la URI absoluta base del documento
document.body	Devuelve el elemento <body>
document.cookie	Devuelve la cookie del documento
document.doctype	Devuelve del tipo de archivo del documento
document.documentElement	Devuelve el elemento <html>
document.documentMode	Devuelve el modo usado por el navegador
document.documentURI	Devuelve la URI absoluta del documento
document.domain	Devuelve el servidor de nombres
document.embeds	Devuelve todos los elementos <embed>
document.forms	Devuelve todos los elementos <form>
document.head	Devuelve el elemento <head>
document.images	Devuelve todos los elementos <img>
document.implementation	Devuelve la implementación DOM
document.inputEncoding	Devuelve al codificación del documento
document.lastModified	Devuelve el día y la hora de modificación del documento
document.links	Devuelve todos los elementos <area> y <a> que tengan definido el atributo <i>href</i>
document.readyState	Devuelve el estado de carga del documento
document.referrer	Devuelve la URI del documento referencia
document.scripts	Devuelve todos los elementos <script>
document.strictErrorChecking	Devuelve si el control de errores está activo
document.title	Devuelve el elemento <title> del documento
document.URL	Devuelve la URL completa del documento



*Algunas de estas propiedades pueden devolver más un elemento, es decir, más de un valor por lo que estos valores deben recogerse en variables de tipo array de una dimensión donde cada componentes es uno de valores.*



- **Acceso a los nodos**

Una vez creado el árbol de nodos DOM, el acceso a cualquiera de ellos se puede hacer o bien a través de sus nodos padre partiendo del nodo raíz o bien accediendo directamente a dicho nodo a través de los métodos y propiedades DOM.

- **Método `getElementsByTagName("etiqueta")`**

Este método devuelve todos los elementos de un documento html que tengan como etiqueta la pasada como parámetro. Como puede haber varios elementos iguales, lo que implica que tienen etiquetas iguales, el resultado se guarda en una variable de tipo array.

**Ejemplo breve:**

```
var divisiones = document.getElementsByTagName("div");
```

En este ejemplo vamos a buscar todos los elementos `<div>` de nuestro documento y guardarlo en el array `divisiones`. Para ellos utilizamos el método `getElementsByTagName()` pasándole como parámetro la etiqueta `"div"` e indicando que busque desde la raíz del documento (`document`). En `divisiones[0]` estará el primer `<div>`, en `divisiones[1]`, el el segundo `<div>` y así sucesivamente.

Ahora podríamos cambiar el lenguaje de cualquier `<div>` (atributo **lang**), por ejemplo, a través de los elementos de `"divisiones"` como podría ser

```
divisiones[0].lang="en";
```

**Ejemplo completo:**

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Título</title>

 <meta charset="utf-8">

</head>

<body>

 <div lang="en"></div>

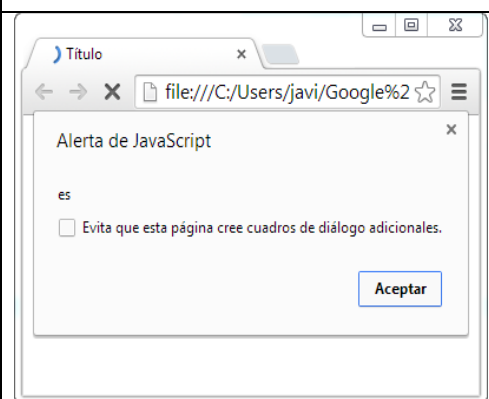
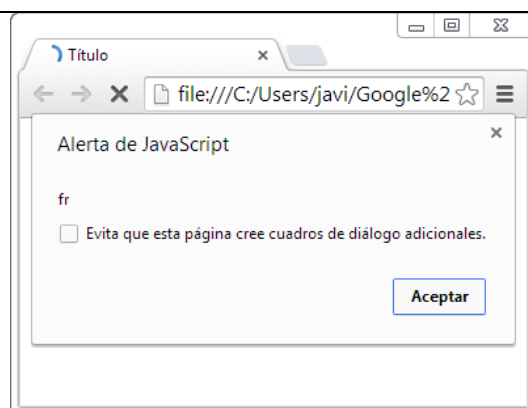
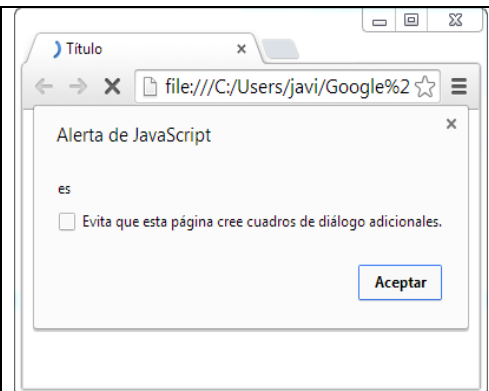
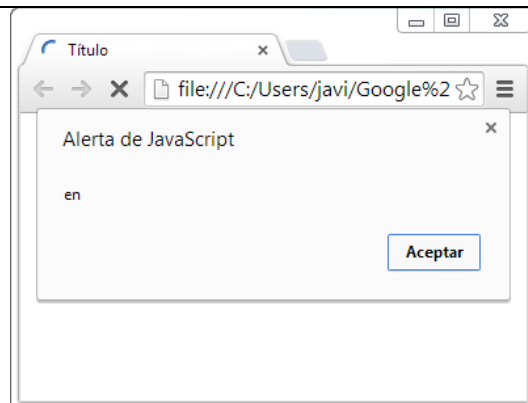
 <div lang="fr"></div>

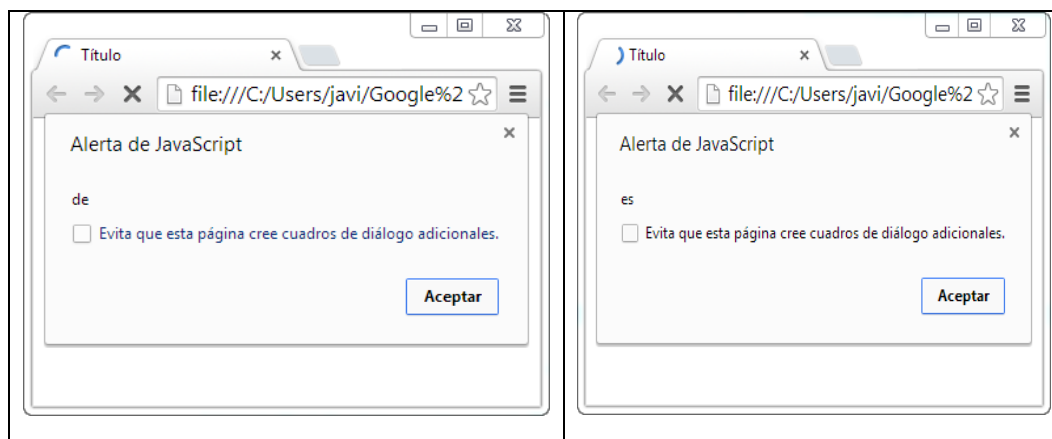
 <div lang="de"></div>
```

```
<script type="text/javascript">
 var divisiones=document.getElementsByTagName("div");
 for (var i=0;i<divisiones.length;i++)
 {
 alert(divisiones[i].lang); //no cambia
 divisiones[i].lang="es";
 alert(divisiones[i].lang);
 }
</script>
```

```
</body>
```

```
</html>
```





En este ejemplo vemos que hay que tres elementos `<div>` cada uno de ellos con un lenguaje asociado (inglés, francés y alemán por este orden). A continuación vemos que está definido el elemento `<script>` dentro del cual vamos a cargar en una variable de tipo array de una dimensión (“divisiones”) cada uno de los elementos `<div>` y, con ayuda de un bucle, vamos a recorrer el vector “divisiones” y a cambiar el valor del atributo “lang” a español (es) en cada uno de los `<div>`. Para que sea más gráfico el script visualiza dos mensajes, el primero con el valor que tiene antes de entrar en el script (imágenes de la columna izquierda) y otro después de haberlo cambiado (imágenes de la columna derecha). Otro punto a tener en cuenta es la estructura del bucle; en él vemos que la variable de control “i” va de 0 a “divisiones.length”-1. La propiedad “length” de cualquier array devuelve el número de elementos por lo que en este caso “divisiones.length” devuelve 3 lo que implica que el bucle tiene que ir de 0 a 2 (“divisiones.length” -1).

- Método **getElementsByName**(“nombre de elemento”)

Busca los elementos del documento cuya atributo “name” sea el parámetro pasado al método. Este atributo normalmente es único por lo que este método viene muy bien para acceder directamente a un elemento concreto

	<i>Hay una excepción clara a lo de la unicidad del atributo <b>name</b> y son los “botones de opción” o “radiobuttons” que comparten el mismo nombre o atributo <b>name</b>. En este caso a la variable a la que se asigne este método será tratada como una array.</i>
-------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### Ejemplo breve:

```
var nombre = document.getElementsByName("nom");
```

En este ejemplo buscamos por todo el documento html los elementos que tenga como atributo **name** el valor “nom” y lo asigna a la variable “nombre”.

Al igual que en el ejemplo anterior y suponiendo que solo hay un elemento con **name="nom"**, a través de la variable "nombre" (más concretamente "nombre[0]" ya que se sigue interpretando como una variable array) podríamos cambiar cualquier atributo del elemento como, por ejemplo, el lenguaje asociado (atributo **lang**).

```
nombre[0].lang="es";
```

### Ejemplo completo:

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Título</title>

 <meta charset="utf-8">

 </head>

 <body>

 <div lang="en" name="uno"></div>

 <div lang="fr" name="dos"></div>

 <div lang="de" name="tres"></div>

 <script type="text/javascript">

 var nombre=document.getElementsByName("dos");

 alert(nombre[0].lang);

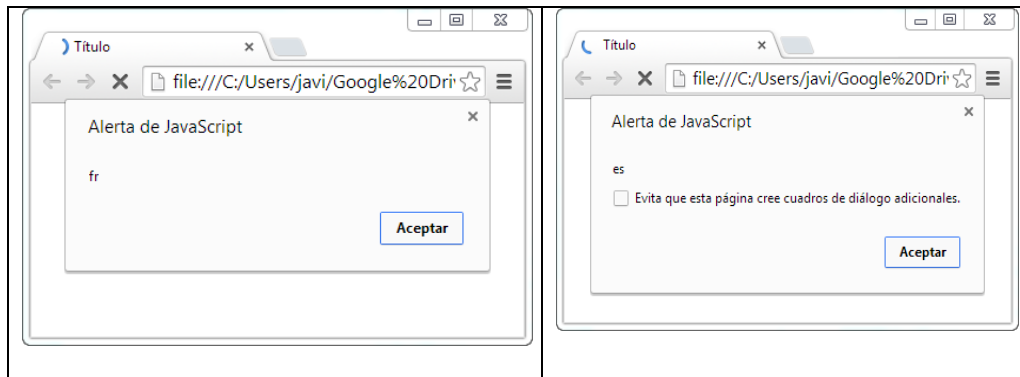
 nombre[0].lang="es";

 alert(nombre[0].lang);

 </script>

 </body>

</html>
```



En este ejemplo tenemos tres divisiones con tres nombres diferentes (uno, dos y tres) y con tres idiomas asociados diferentes (inglés, francés y alemán). Al ejecutar el script asignamos a la variable “nombre” el elemento html cuyo atributo **name** es “dos” (segunda división con idioma asociado francés o “fr”) y vemos que el primer mensaje indica que el lenguaje del elemento es “fr” (imagen columna izquierda) y que, cuando se cambia, su valor es “es” (imagen columna derecha). Resaltar la idea de que nombre sigue siendo una variable array y que el elemento se guardará en la posición “0” del array.

- Método **getElementById(“id del elemento”)**  
Busca el único elemento del documento html cuyo atributo **id** coincida con el valor que se le ha pasado como parámetro.

#### Ejemplo breve:

```
var nombre = document.getElementById("nom");
```

En este ejemplo buscamos el único elemento que haya en el documento cuyo atributo **id** sea “nom” y lo guardamos en la variable “nombre”.

Al igual que en los ejemplos anteriores, a través de la variable “nombre” podríamos cambiar cualquier atributo del elemento como, por ejemplo, el lenguaje asociado (atributo **lang**).

```
nombre.lang="es";
```

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title>Título</title>
```

```

<meta charset="utf-8">

</head>

<body>

 <div lang="en" id="uno"></div>

 <div lang="fr" id="dos"></div>

 <div lang="de" id="tres"></div>

 <script type="text/javascript">

 var nombre=document.getElementById("dos");

 alert(nombre.lang);

 nombre.lang="es";

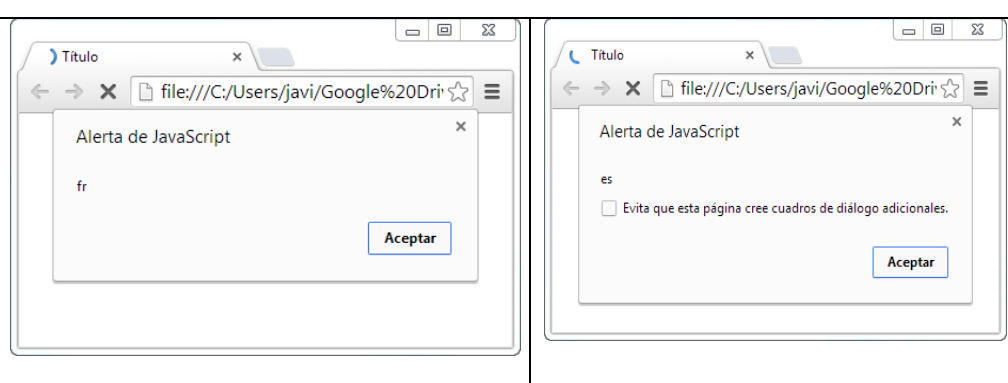
 alert(nombre.lang);

 </script>

</body>

</html>

```



Este ejemplo es muy parecido a los otros dos que hemos visto y la única diferencia es que la variable “nombre” no se toma como un array sino como una variable simple.

- **Creación de nodos**

El proceso de creación de nodos va directamente relacionado con el de agregar nuevos elementos a una página web de forma dinámica, es decir, que dicho elemento no

estaba en el diseño original y se añade cuando sucede algún cambio o se lanza algún evento.

Los cuatro tipos más importantes de nodos de la especificación DOM son:

- **Document**: Nodo raíz del que todos los demás descienden directa o indirectamente.
- **Element**: Nodo que representa cada uno de los elementos HTML5. Puede contener atributos y otros nodos pueden descender de él.
- **Attr**: Nodo que representa un atributo de un elemento HTML5
- **Text**: Nodo que contiene el texto que va dentro de un elemento HTML5.

Los cuatro pasos que hay que seguir cuando se crea un nuevo elemento a un documento son:

1. Creación de un nodo de tipo **Element** que va a ser, en realidad, el elemento que queremos añadir.
2. Creación de un nodo de tipo **Text** que va a tener el contenido del elemento (normalmente es un texto)
3. Se añade el nodo **Text** como nodo hijo del nodo **Element**
4. Se añade el nodo **Element** al documento html como hijo o bien de otro elemento o bien de <body> o <head> dependiendo del tipo de elemento que sea.

### Ejemplo básico de la creación de un artículo con el texto “Artículo”

```
/* Crear un nodo de tipo Element */

var articulo=document.createElement("article");

/*Crear un nodo de tipo Text */

var texto=document.createTextNode("Artículo");

/*Añadir el nodo Text como hijo de Element */

articulo.appendChild(texto);

/*Añadir el nodo Element como elemento de <body> */
```

```
document.body.appendChild(articulo);
```

**Ejemplo completo:**

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Título</title>

 <meta charset="utf-8">

 <style>

 article {background-color:orange;}

 </style>

 </head>

 <body>

 <script type="text/javascript">

 var articulo = document.createElement("article");

 var texto = document.createTextNode("Artículo");

 articulo.appendChild(texto);

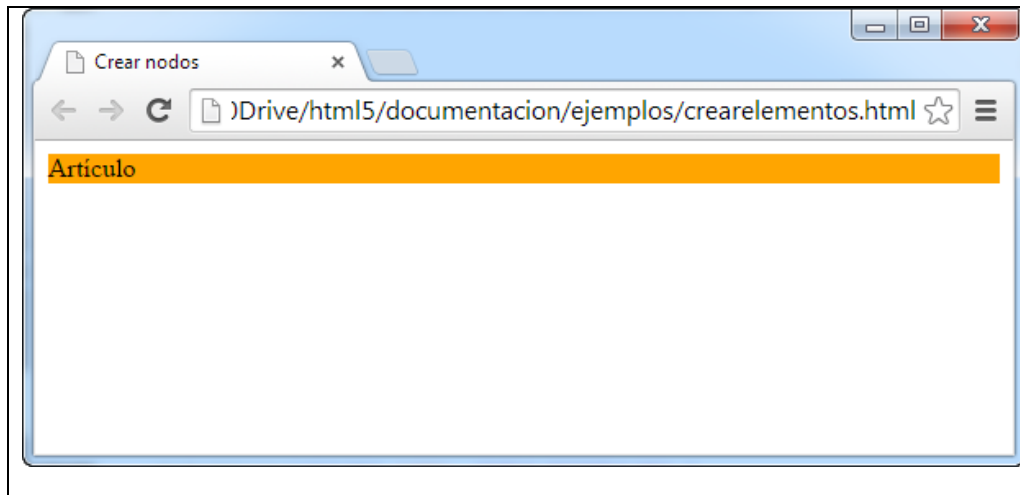
 document.body.appendChild(articulo);

 </script>

 </body>

</html>
```





En este ejemplo se ve claramente que el elemento `<article>` es creado por el script de la página html y que se comporta como un elemento normal ya que, por ejemplo, se le pueden asignar estilos sin ningún problema.

- **Eliminación de nodos**

La eliminación de nodos consta de dos pasos: búsqueda de elemento a través de los métodos que hemos visto anteriormente y borrado del elemento con la función **`removeChild("nodo a borrar")`**.

**Ejemplo básico de borrado del artículo con `id="articulo1"`**

```
var articulo = document.getElementById("articulo1");
articulo.parentNode.removeChild(articulo);
```

En este caso buscamos el elemento cuyo **id** es "articulo1" y lo guardamos en la variable "articulo". La siguiente instrucción

```
articulo.parentNode.removeChild(articulo);
```

se interpreta de la siguiente manera:

`articulo.parentNode` es una propiedad que nos permite saber el padre de un elemento (en este caso el elemento es "articulo"). Averiguado el padre se le aplica la función **`removeChild(articulo)`** para borrar ese hijo del nodo padre.

	<i>Si borras un nodo eliminas todos sus hijos a la vez por lo que no hay que ir uno por uno para quitarlos todos</i>
-------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------

### Ejemplo completo de borrado del artículo con id="articulo1"

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Borrar nodos</title>

 <meta charset="utf-8">

 <style>

 #articulo1

 { background-color:orange;}

 .fondoverde {background-color:#ccffcc;}

 #articulo2 {background-color: grey;}

 </style>

 </head>

 <body class="fondoverde">

 <article id="articulo1">

 <h1> Artículo 1</h1>

 </article>

 <article id="articulo2">

 <h1> Artículo 2</h1>

 </article>

 <script type="text/javascript">

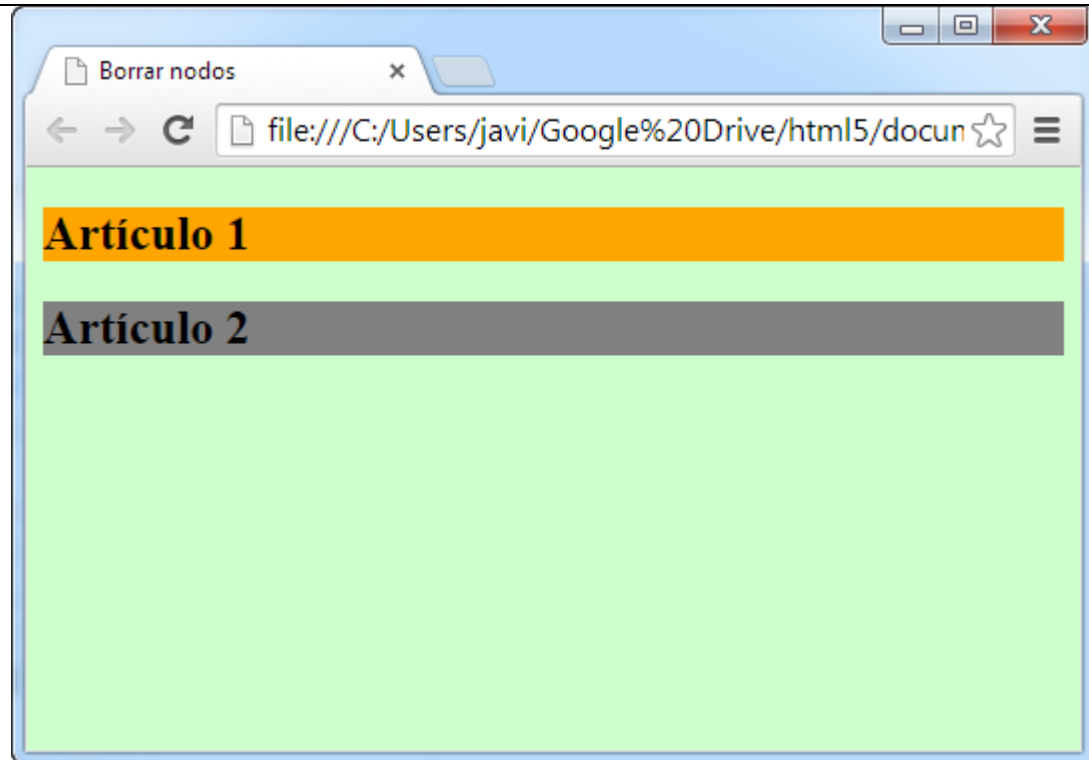
 var articulo=document.getElementById("articulo1");

 articulo.parentNode.removeChild(articulo);

 </script>
```

```
</body>
```

```
</html>
```



En este ejemplo vemos como estaría la página html si no tuviera el script de borrado de un elemento (primera imagen) y luego como quedaría con dicho script (segunda imagen). En esta segunda imagen podemos observar que al borrar el elemento `<articulo>` con `id="articulo1"` hemos borrado también el elemento `<h1>` que dependía de él.

- **Reemplazar nodos**

Con el método **`replaceChild("nodo nuevo", "nodo viejo")`** podemos cambiar un elemento HTML5 por otro de forma dinámica.

**Ejemplo básico de cambio de una artículo con `id="articulo1"` por un párrafo con el texto "Párrafo uno"**

```
var viejo = document.getElementById("articulo1");

var nuevo = document.createElement("p");

var texto = document.createTextNode("Párrafo uno");

nuevo.appendChild(texto);

viejo.parentNode.replaceChild(nuevo,viejo);
```

Para buscar el nodo padre del elemento que queremos reemplazar utilizamos la misma forma que en el borrado de nodos y vemos que javascript permite reemplazar un elemento por otro diferente.

**Ejemplo completo:**

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Borrar nodos</title>

 <meta charset="utf-8">

 <style>

 #articulo1

 { background-color:orange;}

 .fondoverde {background-color:#ccffcc;}

 </style>

 </head>

</html>
```

```
#articulo2 {background-color: grey;}

p {background-color:#999ff;}

</style>
</head>
<body class="fondoverde">

 <article id="articulo1">

 <h1> Artículo 1</h1>

 </article>

 <article id="articulo2">

 <h1> Artículo 2</h1>

 </article>

 <script type="text/javascript">

 var viejo = document.getElementById("articulo1");
 var nuevo = document.createElement("p");
 var texto = document.createTextNode("Párrafo1");
 nuevo.appendChild(texto);
 viejo.parentNode.replaceChild(nuevo,viejo);

 </script>

</body>
</html>
```



En este ejemplo si no estuviera el script se vería la primera imagen y con el script cambiamos el elemento `<article>` con `id="articulo1"` por un elemento `<p>` con el texto "Párrafo1" (segunda imagen).

- **Escribir un texto directamente en la página web**

Con el método **write("texto")** aplicado al objeto **document** podemos insertar texto en nuestras páginas de forma dinámica.

**Ejemplo:**

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title>Borrar nodos</title>

 <meta charset="utf-8">

 <style>

 .fondoverde {background-color:#ccffcc;}

 </style>

 </head>

 <body class="fondoverde">

 Inicio

 <script type="text/javascript">

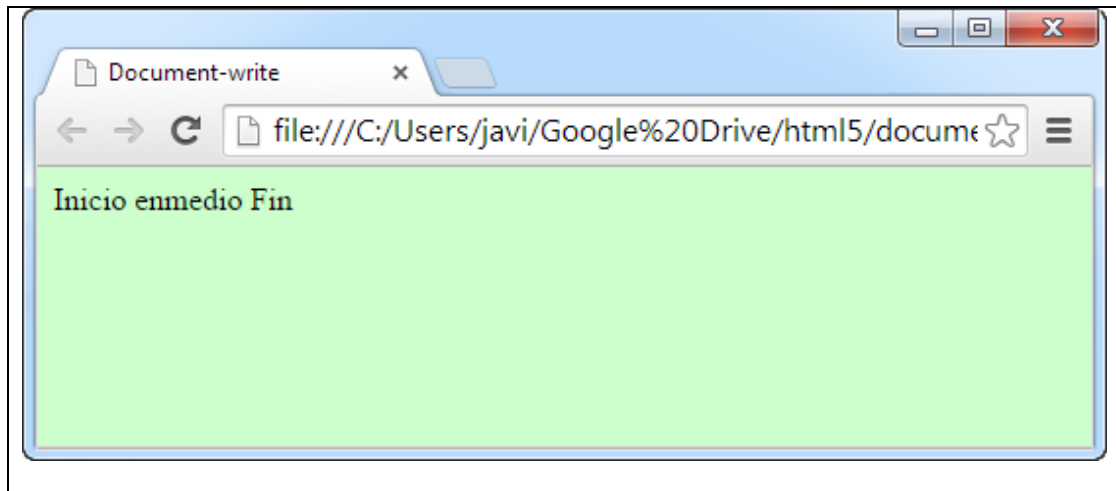
 document.write("En medio");

 </script>

 Final

 </body>

</html>
```



En este ejemplo se escribe primero “Inicio”, luego el script escribe “En medio” y por último “Fin”.

- **Cambiar el contenido de un elemento**

A través de la propiedad **innerHTML** podemos cambiar el contenido de un elemento.

**Ejemplo básico de cambio de texto de un elemento con id="p1"**

```
document.getElementById("p1").innerHTML = "Nuevo texto";
```

En este caso se cambiaría el texto que tuviera el elemento con id="p1" y tendría el texto “Nuevo texto”

**Ejemplo completo de cambio de texto de un elemento con id="p1":**

```
<!DOCTYPE html>
<html lang="es-es">
 <head>
 <title>Cambio contenido</title>
 <meta charset="utf-8">
 <style>
 .fondoverde {background-color:#ccffcc;}
 p
 { background-color:orange;}
 </style>
```



```
</head>

<body class="fondoverde">

 <p id="p1"> Texto inicial </p>

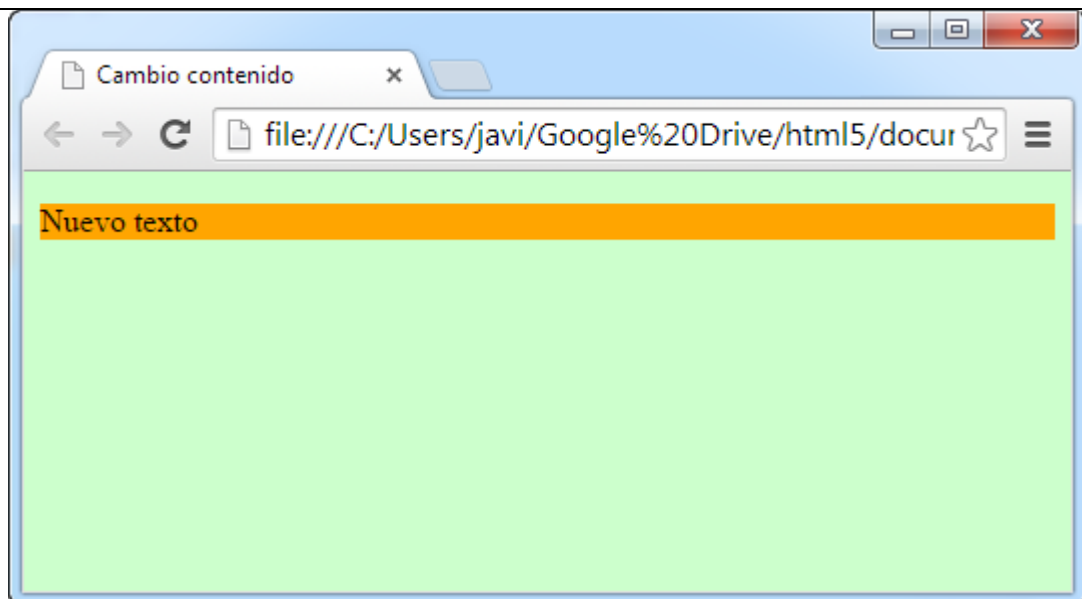
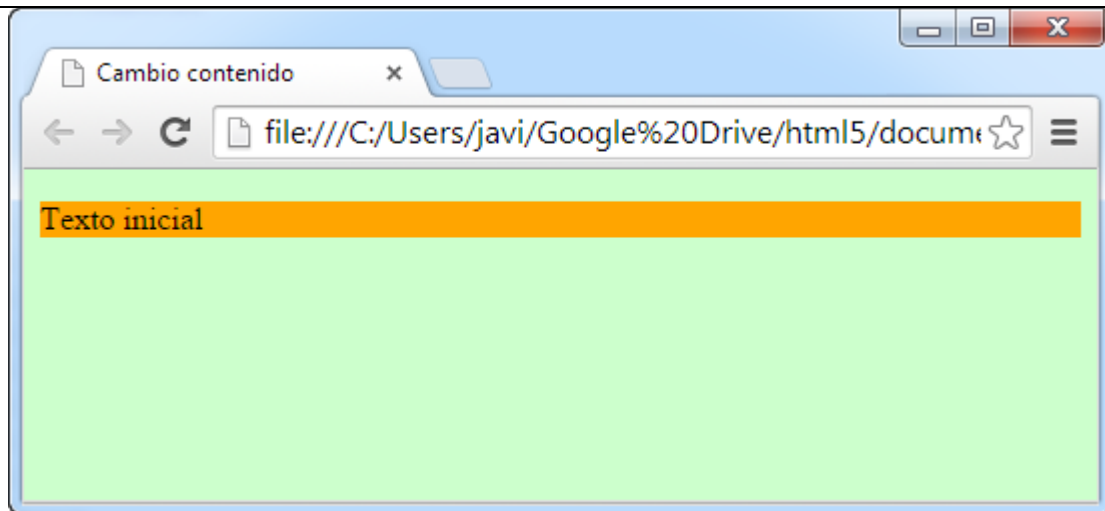
 <script type="text/javascript">

 document.getElementById("p1").innerHTML = "Nuevo texto";

 </script>

</body>

</html>
```



En la primera imagen del ejemplo vemos cómo quedaría la página si tuviera el script y en la segunda es como se vería con el script de cambio de contenido del elemento cuyo id="p1"

- **Cambio del valor de un atributo de un elemento html**

Una posible forma para cambiar un atributo de un elemento html es la siguiente:

```
document.getElementById("id").atributo="valor";
```

**Ejemplo breve del cambio de lenguaje a español del elemento con id="p1":**

```
document.getElementById("p1").lang="es"
```

- **Cambio del estilo de un elemento html**

El HTML DOM permite a Javascript cambiar los estilos de los elementos html. Una posible forma es la siguiente:

```
document.getElementById("p1").style.propiedad="nuevo estilo"
```

**Ejemplo breve del cambio de color del fondo a naranja de un elemento con id="p1"**

```
document.getElementById("p1").style.background-color="orange"
```



*En todos estos ejemplos se ha usado una forma muy poco práctica de insertar scripts Javascript ya que lo normal es usar un archivo externo que se agrega a <head> compuesto de funciones que se llaman a través de eventos*

## 7.6. EVENTOS

Un evento es una acción como puede ser un clic de ratón, una tecla pulsada, etc., que recoge un script o programa y, en base a esa acción, el programa realiza unas determinadas tareas.

Los eventos permiten al usuario interactuar con los elementos de cualquier programa e, incluso, pasar información por lo que es muy importante saberlos controlar para dotar de un mejor dinamismo a nuestras páginas web.

	<i>JavaScript es un <b>lenguaje basado en eventos</b>, es decir, que los programas escritos en este lenguaje pueden cambiar su comportamiento en base a los eventos que se produzcan.</i>
-----------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Ejemplos de eventos en una documento html son:

- Cuando el usuario hace clic con el ratón sobre un determinado elemento.
- Cuando una página web se carga.
- Cuando una imagen ha sido descargada
- Cuando se mueve el ratón sobre un elemento.
- Cuando un campo de formulario input se cambia.
- Cuando un formulario se envía.
- Cuando el usuario pulsa una tecla.

	<i>JavaScript permite asociar funciones a cada uno de los eventos que se producen.</i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------

- **Tipos de eventos**

Un mismo tipo de evento se puede aplicar sobre varios elementos html al igual que un elemento puede soportar varios tipos de eventos. Por ejemplo, hacer clic con el botón izquierdo del ratón se puede realizar sobre cualquier elemento HTML5, en cambio, cuando nos salimos de una página web el evento, denominado **onunload**, solo se puede asociar al elemento <body>.

La lista de eventos es muy extensa por lo aquí tienes un listado de los más importantes:

### Eventos de ratón:

Evento	Descripción
onclick	El evento se produce cuando el usuario hace clic sobre un elemento.
ondblclick	El evento se produce cuando el usuario hace doble-clic sobre un elemento
onmousemove	El evento se produce cuando el puntero se mueve mientras está encima del elemento
onmouseenter	El evento se produce cuando el puntero entra en el elemento
onmouseleave	El evento se produce cuando el puntero sale del elemento
onmousedown	El evento se produce cuando se pulsa sin soltar un botón del ratón

### Eventos de teclado:

Evento	Descripción
onkeydown	El evento se produce cuando el usuario presiona una tecla sin soltarla
onkeypress	El evento se produce cuando el usuario presiona una tecla
onkeyup	El evento se produce cuando el usuario suelta una tecla pulsada

### Eventos de marcos/objetos:

Evento	Descripción
onload	El evento se produce cuando se carga un objeto
onresize	El evento se produce cuando se redimensiona la vista del documento
onunload	El evento se produce cuando se sale de la página (solo para <body>)

**Eventos de formulario:**

Evento	Descripción
onblur	El evento se produce cuando se deselecciona un elemento
onchange	El evento se produce cuando se cambia el contenido
onfocus	El evento se produce cuando el elemento está seleccionado
onreset	El evento se produce cuando el formulario se resetea
onselect	El evento se produce cuando el usuario selecciona un texto
onsubmit	El evento se produce cuando el formulario es enviado

**Ejemplos:**

- En este primer ejemplo se va a visualizar una ventana con el mensaje “Has pulsado el botón” cuando se hace un clic de ratón sobre un botón de un formulario

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title> Pulsar un botón </title>

 <meta charset="UTF-8">

 <style>

 body { background-color:#ccffcc; }

 </style>

 <script>

 function pulsar()

 { alert("Has pulsado el botón"); }

 </script>

 </head>
```

```
<body>

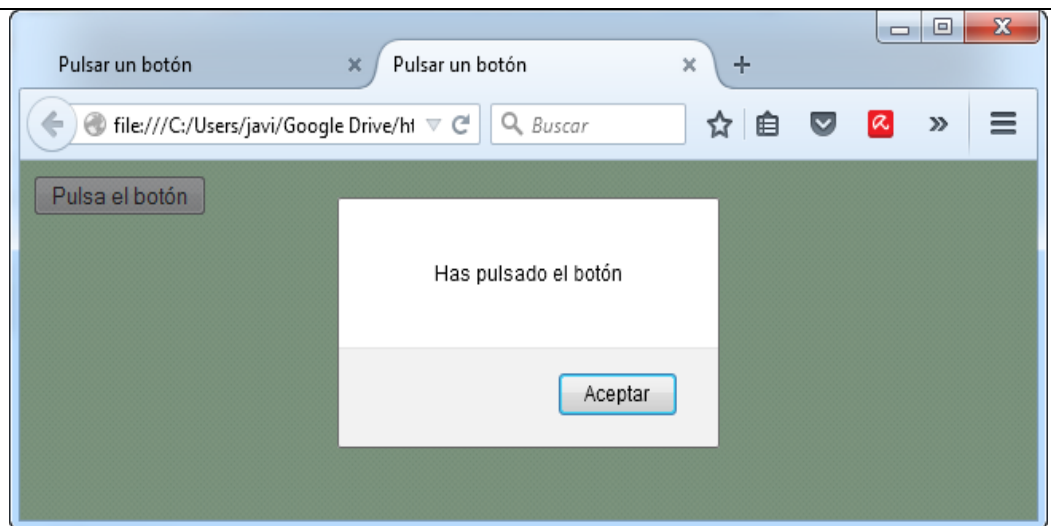
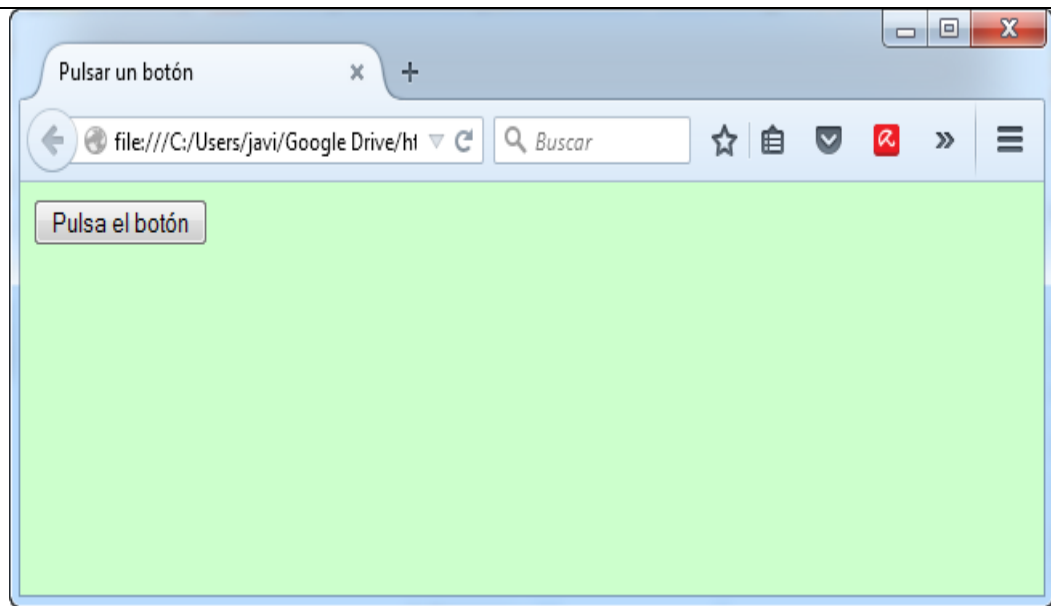
 <form>

 <input type="button" value="Pulsa el botón" onclick="pulsa()">

 </form>

</body>

</html>
```



Vemos en este ejemplo que se ha asociado un evento “onclick” al elemento “button” de un formulario, es decir, cuando se pulsa dicho elemento se ejecuta el código de la función “pulsa()” que abre una ventana, comando *alert*, con un mensaje.

- No es estrictamente necesario utilizar elementos de formulario para asociarles eventos. En el siguiente ejemplo vamos a aplicar el evento “onclick” a un párrafo o elemento <p>

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title> Pulsar sobre el párrafo </title>

 <meta charset="UTF-8">

 <style>

 body { background-color:#ccffcc; }

 p { background-color:yellow;
 width: 90px;
 height: 60px; }

 </style>

 <script>
 function pulsar()
 { alert("Has pulsado sobre el párrafo");
 }
 </script>

 </head>

 <body>

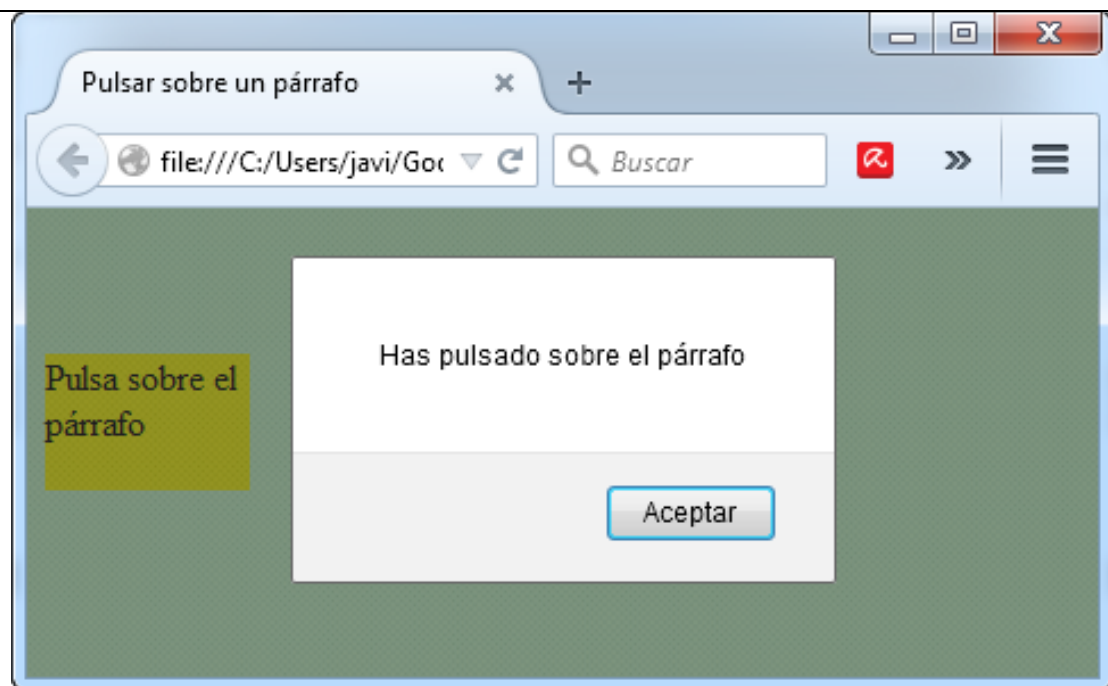
 <p onclick="pulsar()">
```

Pulsa sobre el párrafo

</p>

</body>

</html>





En este ejemplo al pulsar el botón izquierdo del ratón sobre el párrafo de fondo amarillo, se ejecuta el código de la función “pulsar()” que abre una ventana con el mensaje “Has pulsado sobre el párrafo”.

- Ahora vamos a utilizar eventos para cambiar las propiedades de los elementos de nuestras páginas a través de lo visto en el tema anterior, DOM, En el siguiente ejemplo tenemos dos botones que, cuando se pulse sobre ellos, van a cambiar el color de fondo de un párrafo.

```
<!DOCTYPE html>

<html lang="es-es">

<head>

 <title> Cambiando de color </title>

 <meta charset="UTF-8">

 <style>

 body { background-color:#ccffcc; }

 p { background-color:yellow;

 width: 200px;

 height: 60px; }

 </style>

 <script>
```

```
function azul()
{
 document.getElementById("p1").style.background= "blue";
}
```

```
function verde()
```

```
{
 document.getElementById("p1").style.background= "green";
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<form>
```

```
<input type="button" value="azul" onclick="azul()">
```

```
<input type="button" value="verde" onclick="verde()">
```

```



```

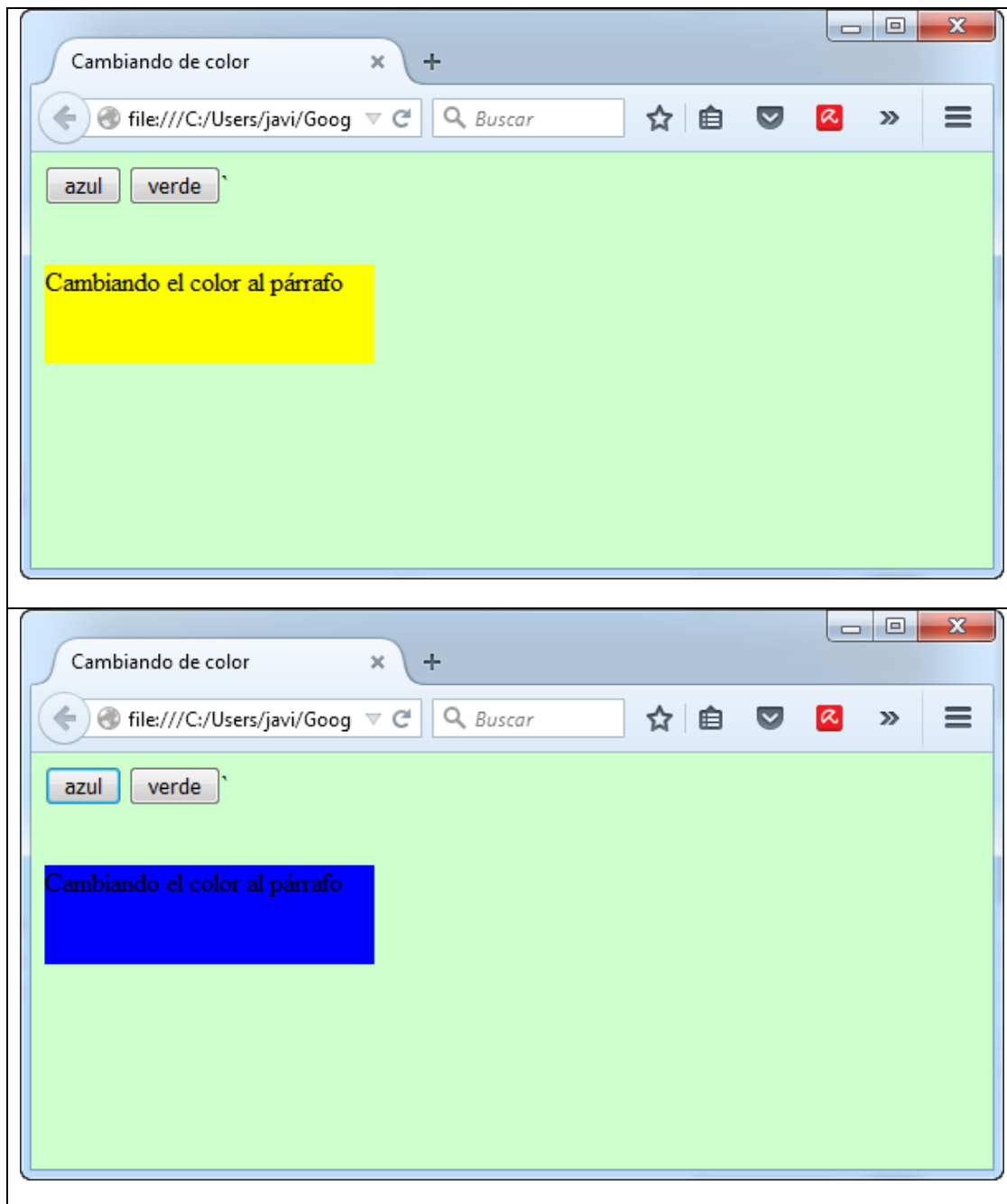
```
<p id="p1">
```

Cambiando el color al párrafo

```
</p>
```

```
</body>
```

```
</html>
```





En la primera imagen se ven los dos botones y el párrafo con color amarillo de fondo. Al pulsar el botón con el texto “azul” , el color de fondo del párrafo pasa a ser azul debido a que se ejecuta la función “azul” (segunda imagen). También se puede pulsar el botón verde y cambiar el color de fondo del párrafo a verde a través del código de la función “verde”. Para hacer este cambio de color accedemos al párrafo a través de su “id” que en el ejemplo es “p1”

- En el siguiente ejemplo vamos a utilizar el evento “onmousemove” para cambiar el color de fondo de un párrafo cuando pasa el ratón por encima de él

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title> Ratón sobre elemento </title>

 <meta charset="UTF-8">

 <style>

 body { background-color:#ccffcc; }

 p { background-color:yellow;

 width: 200px;

 height: 60px; }
```

```
</style>
```

```
<script>
```

```
function mover()
{
 document.getElementById("párrafo").style.background=
 "red";
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```



```

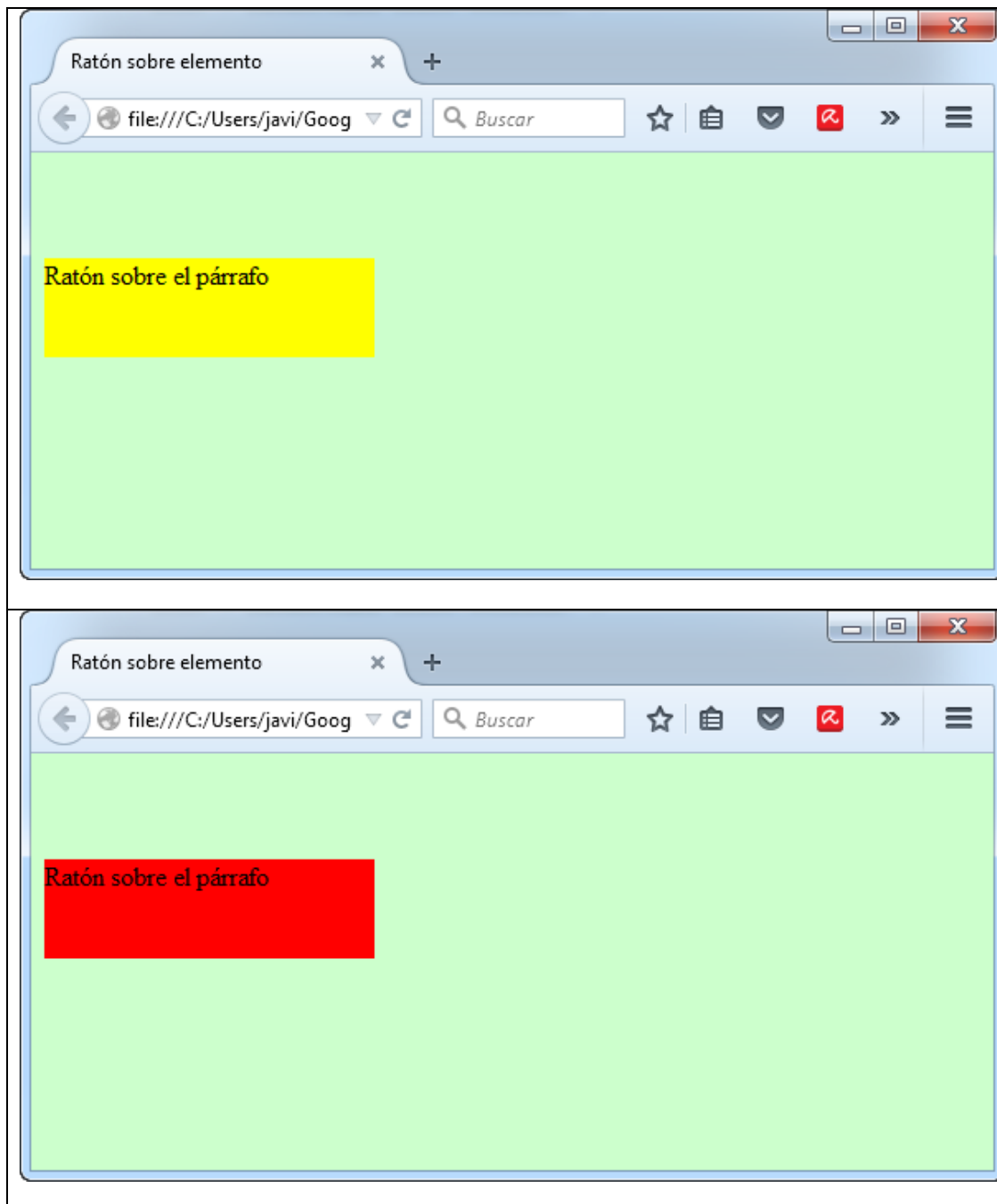
```
<p onmousemove="mover()" id="párrafo">
```

Ratón sobre elemento

```
</p>
```

```
</body>
```

```
</html>
```



En este ejemplo, cuando el ratón pasa por encima del párrafo con id “párrafo” cambia su color de fondo de amarillo a rojo.

- También podemos manipular elementos html con los métodos DOM vistos en el tema anterior y los eventos en tiempo real. En este ejemplo vamos a cambiar el texto de un párrafo cuando se pulsa un botón.

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title> Cambio contenido </title>

 <meta charset="UTF-8">

 <style>

 body { background-color:#ccffcc; }

 p { background-color:yellow;
 width: 200px;
 height: 60px; }

 </style>

 <script>

 function cambiar()
 { document.getElementById("párrafo").innerHTML=
 "Cambiando el texto"; }

 </script>

 </head>

 <body>

 <form>
```

```
<input type="button" value="pulsa" onclick="cambiar()">
```

```
</form>
```

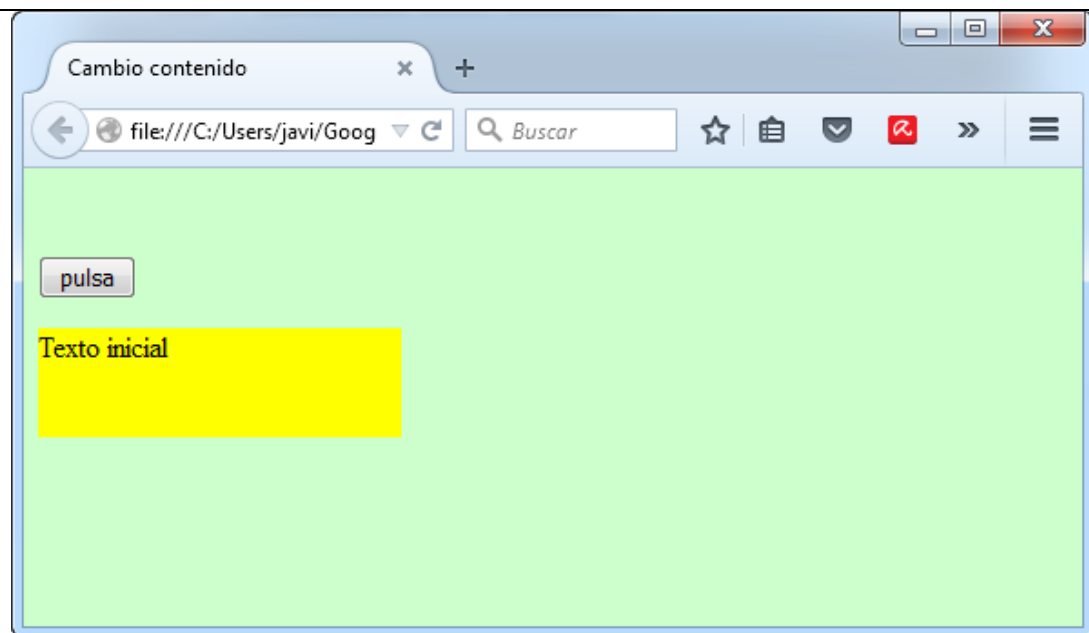
```
<p id="párrafo">
```

Texto inicial

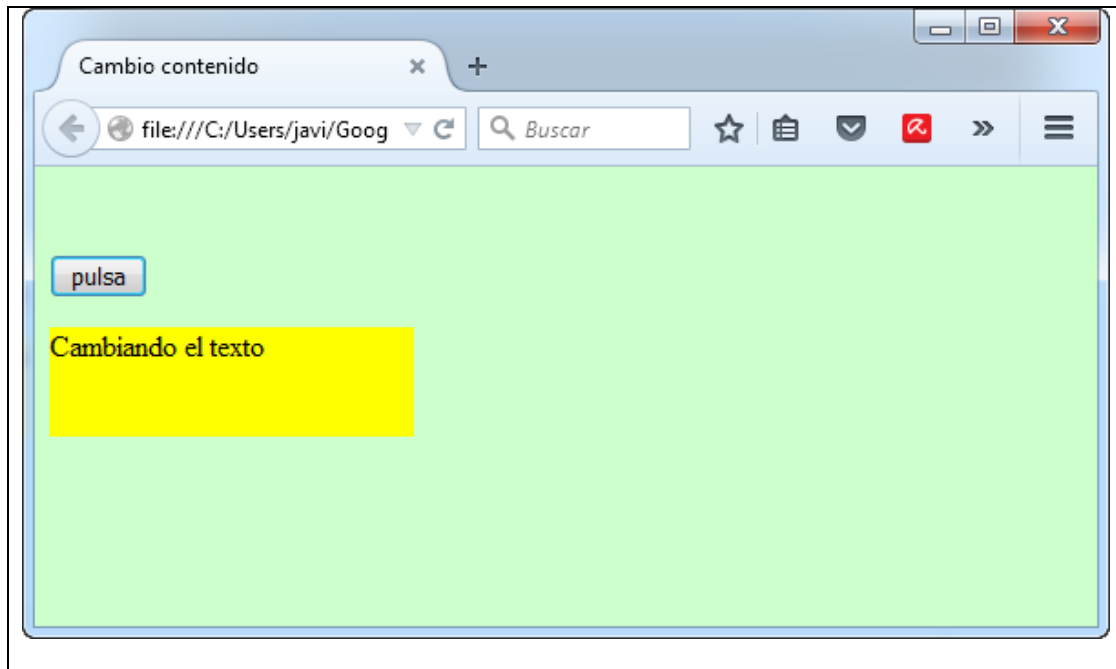
```
</p>
```

```
</body>
```

```
</html>
```







En este ejemplo, cuando se pulsa sobre el botón (evento onclick), se cambia el texto que hay dentro del párrafo con id "párrafo"

- Ahora vamos a crear elementos a través de la aplicación de eventos y del DOM

```
<!DOCTYPE html>

<html lang="es-es">

 <head>

 <title> Creación párrafo </title>

 <meta charset="UTF-8">

 <style>

 body { background-color:#ccffcc; }

 p { background-color:yellow;

 width: 200px;

 height: 60px; }

 </style>
```

```
<script>
```

```
function crear()
{
 var párrafo = document.createElement("p");
 var texto = document.createTextNode("Texto interior");
 párrafo.appendChild(texto); document.body.appendChild(párrafo);
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```



```

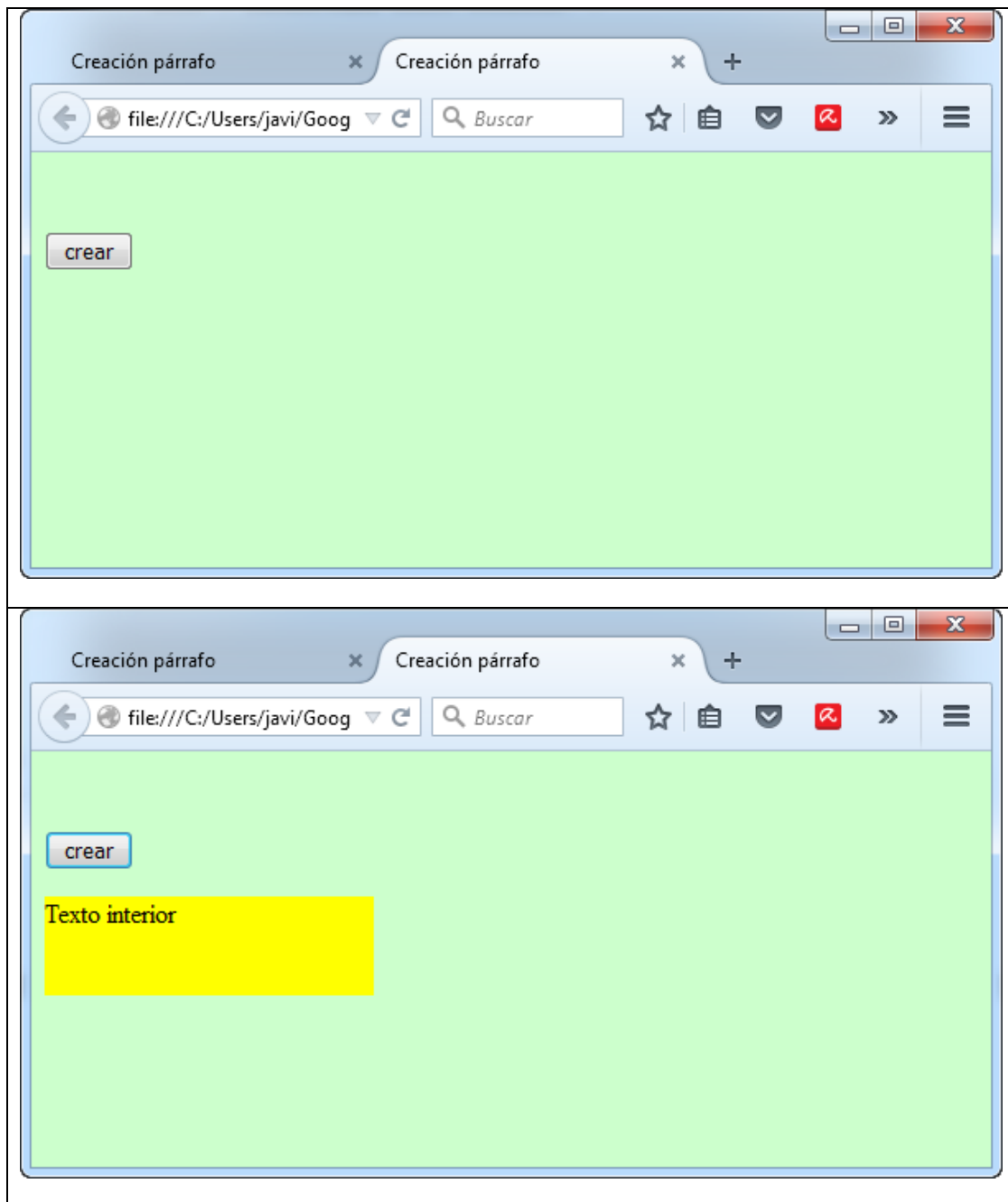
```
<form>
```

```
<input type="button" value="pulsa" onclick="crear()">
```

```
</form>
```

```
</body>
```

```
</html>
```



En este ejemplo, cuando se pulsa el botón, se crea un párrafo con el texto “Texto interior” de la misma forma que hemos visto en el tema sobre DOM. Al ser el elemento “p” un elemento de tipo bloque se crea debajo del formulario y se le aplican los estilos como a cualquier otro elemento de la página.

- A continuación vamos a ver un ejemplo de cómo crear un elemento con un “id” concreto y como borrarlo a través de dos botones. Tomaremos de base el ejemplo anterior.

```
<!DOCTYPE html>

<html lang="es-es">
```

```
<head>

<title> Creación y borrado de párrafo </title>

<meta charset="UTF-8">

<style>

 body { background-color:#ccffcc; }

 p { background-color:yellow;

 width: 200px;

 height: 60px; }

</style>

<script>

function crear()

 {

var párrafo = document.createElement("p");

var texto = document.createTextNode("Texto interior");

párrafo.appendChild(texto);

párrafo.id="par";

document.body.appendChild(párrafo);

 }

function borrar()

 {

var párrafo = document.getElementById("par");

párrafo.parentNode.removeChild(párrafo);

 }
```

```
</script>
```

```
</head>
```

```
<body>
```

```



```

```
<form>
```

```
<input type="button" value="crear" onclick="crear()">
```

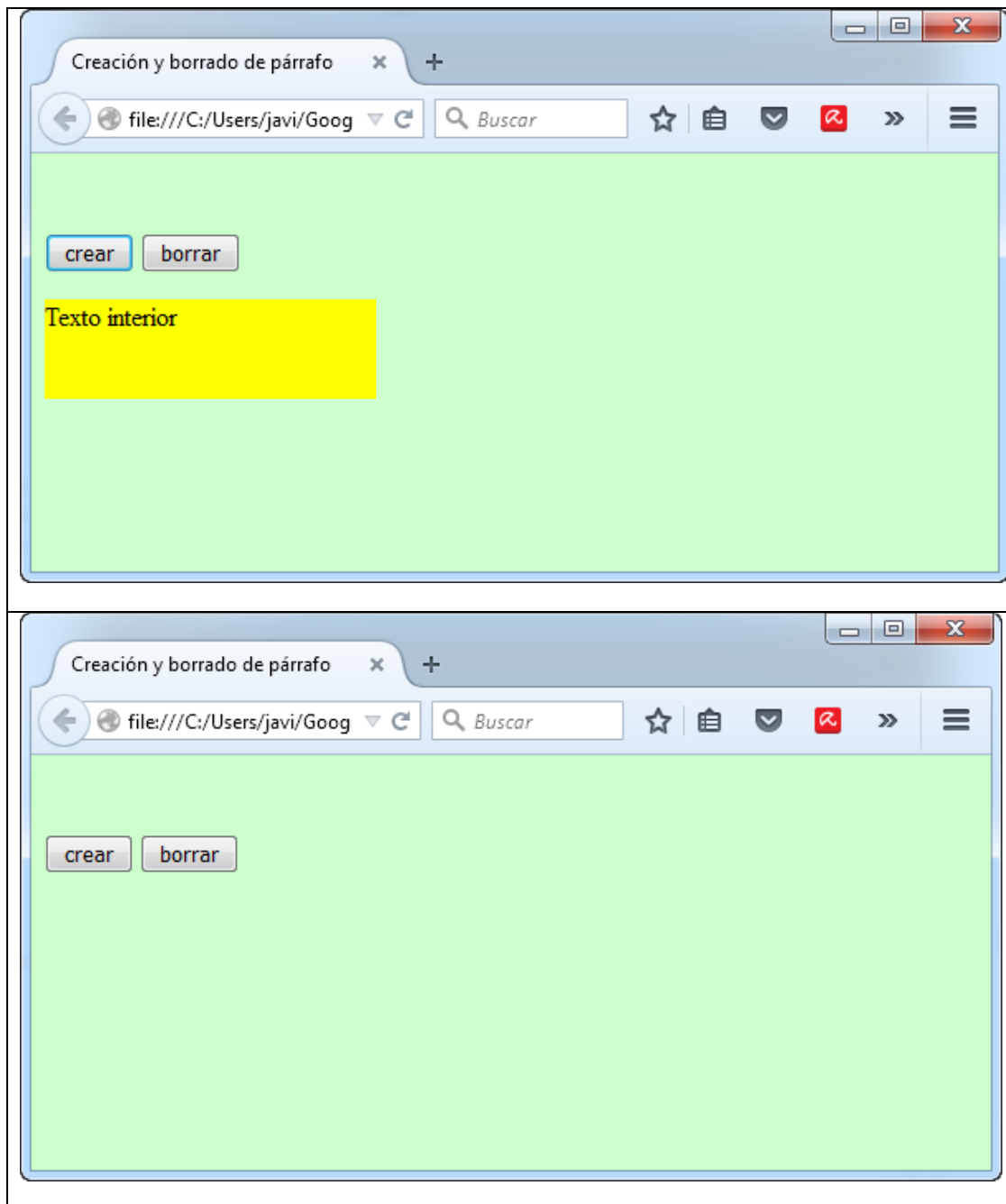
```
<input type="button" value="borrar" onclick="borrar()">
```

```
</form>
```

```
</body>
```

```
</html>
```





En este ejemplo disponemos de dos botones, uno para crear el elemento `<p>` con `id="par"` y otro botón para borrar ese mismo elemento. La segunda imagen refleja la creación del elemento `<p>` y la tercera el borrado del mismo elemento.

- En el siguiente ejemplo vamos a ver cómo ejecutar funciones cuando se carga una página sin necesidad de botones.

```
<!DOCTYPE html>

<html lang="es-es">

 <head>
```

```
<title> Al cargar la página </title>

<meta charset="UTF-8">

<style>

 body { background-color:#ccffcc; }

 p { background-color:yellow;
 width: 200px;
 height: 60px; }

</style>

<script>

 function cargar()
 {
 var párrafo = document.createElement("p");
 var texto = document.createTextNode("Texto interior");
 párrafo.appendChild(texto);
 párrafo.id="par";
 document.body.appendChild(párrafo);
 }

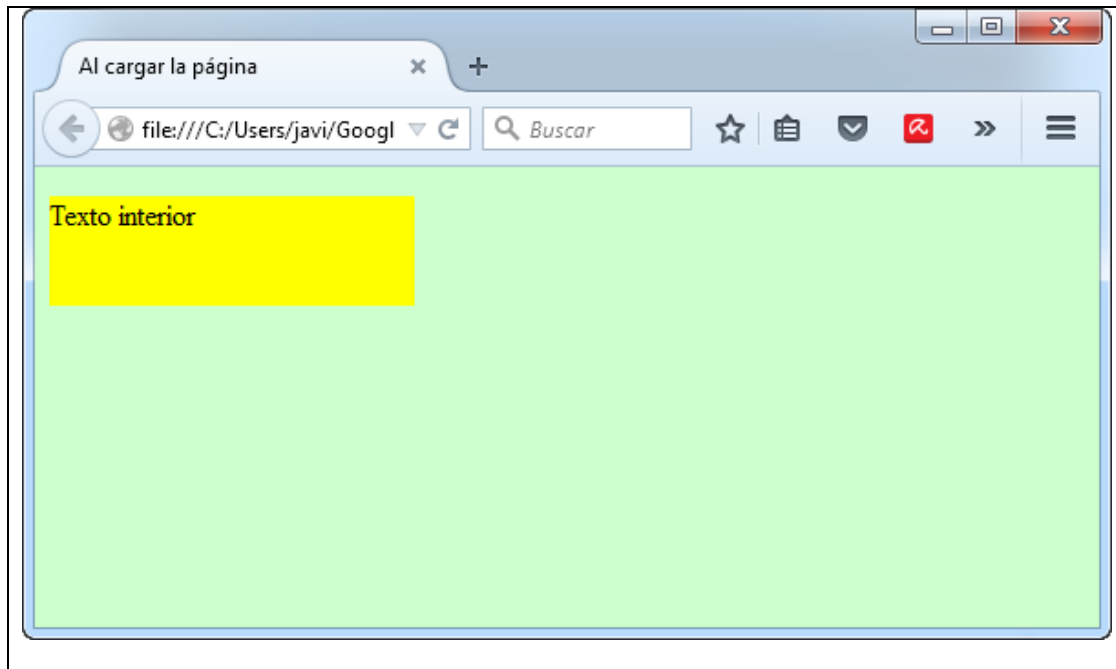
</script>

</head>

<body onload="cargar()">

</body>

</html>
```



En este ejemplo se ve que la parte `<body> ... </body>` no contiene ningún texto ni ningún elemento HTML pero, al cargar la página, se produce el evento "onload" que, asociado a `<body>`, crea un párrafo o elemento `<p>` con `id="par"` y "Texto interior" como texto de dicho párrafo.



## **BLOQUE 8:**

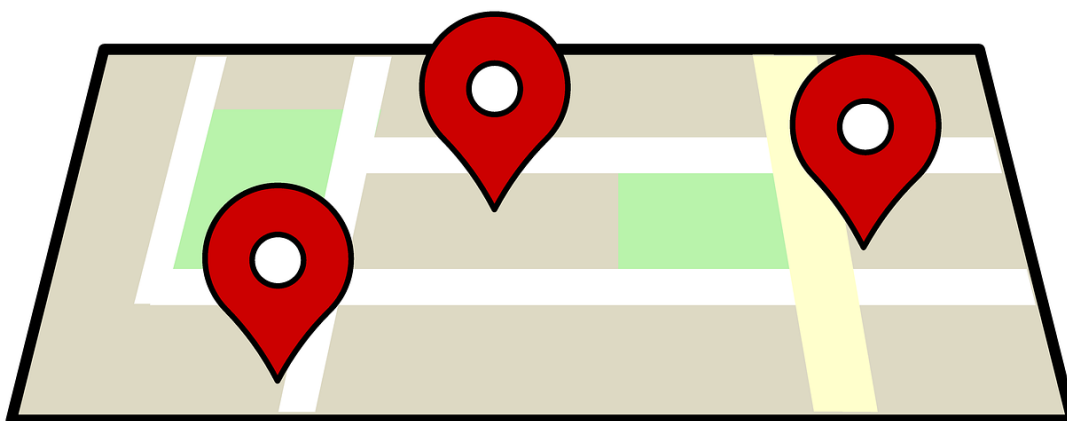
### **HTML5 AVANZADO**

## 8. HTML5 AVANZADO

Las tecnologías HTML5 junto con Javascript marcan uno de los caminos de evolución de las aplicaciones web 2.0, es decir, influyen de forma muy determinante sobre el futuro del diseño y desarrollo web. Un paso muy importante que ha permitido esta influencia es la implementación de APIs (funciones de programación específicas que realizan una determina tarea) por parte de los navegadores modernos lo que ha permitido a los desarrolladores web ahorrar tiempo y simplificar sus aplicaciones ya que sólo se tienen que preocupar en utilizar estos códigos en vez de tenerlos que programar.



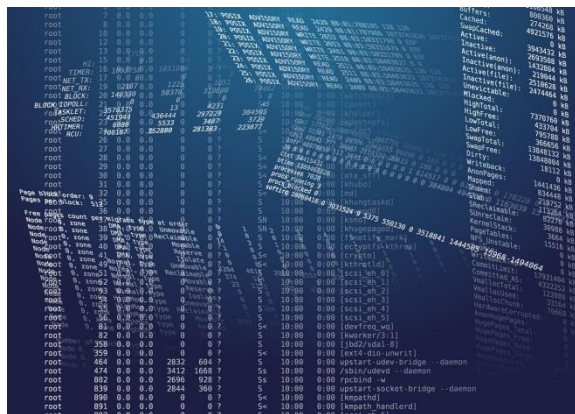
Existen infinidad de APIs y de aplicaciones que usan HTML5 y Javascript por lo que en este tema vamos a ver solamente tres de ellas que nos van a ayudar a hacernos una idea de cómo funcionan y cómo se pueden utilizar.



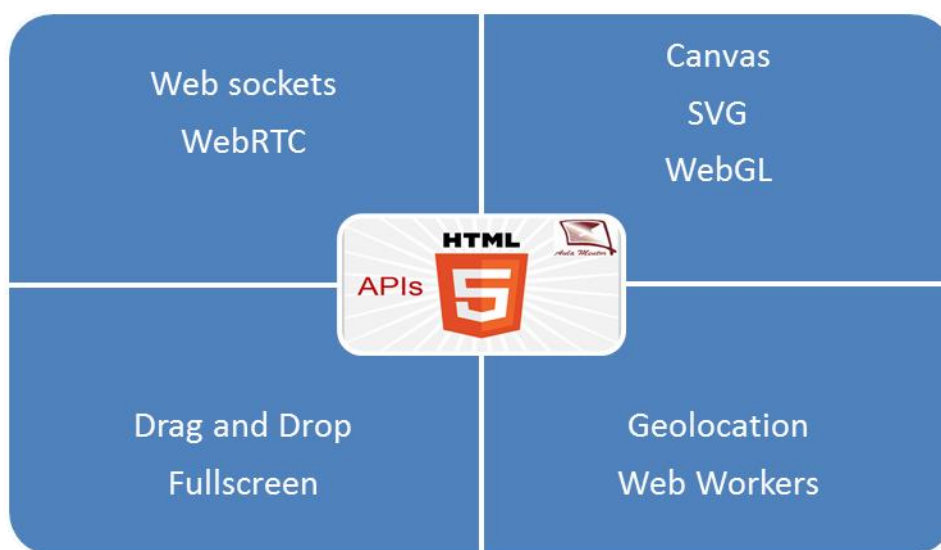
## 8.1. INTRODUCCIÓN

De las nuevas funcionalidades avanzadas importantes que ofrece HTML5 podemos destacar el uso de APIs ya implementadas en los navegadores a través de scripts javascript normalmente. Esto permite aumentar las prestaciones de las páginas web sin tener que recurrir a plugins.

Sin entrar en muchos detalles técnicos se define como API, interfaz de programación de aplicaciones, a las funciones o métodos (programación orientada a objetos) que utiliza otro software (programa, página web, etc.) para realizar una determinada tarea. Esto evita que el programador de ese software tenga que implementar o programar esas funciones que ofrecen las APIs y solamente las utilice sin conocer sus detalles internos.



## Ejemplos de APIs



- **Geolocation**  
Se emplea para conocer la posición geográfica del usuario.
- **Drag and Drop**  
Se emplea para realizar la operación de coger un objeto, trasladarlo y dejarlo en otro lugar.
- **Local Storage**  
Se emplea para crear un espacio de almacenamiento local en el cual se guardan datos. Es la opción que ofrece HTML5 para suplir las “cookies”.

- **Application Cache**  
Se emplea para guardar una copia de la aplicación web en una memoria intermedia o caché y poder emplearla sin tener conexión a Internet.
- **Web Workers**  
Se emplea para ejecutar scripts en una página HTML5 en segundo plano o background.
- **Server-Sent Events**  
Se emplea para actualizar páginas web desde un servidor.
- **Canvas**  
Se emplea para dibujar gráficos en tiempo real a través de scripts javascript.

## 8.2. GEOLOCATION

Esta API se emplea para determinar la posición geográfica de un usuario devolviendo su longitud y latitud.

API					
<b>geolocation</b>	9.0	5.0	3.5	5.0	16

El método **getCurrentPosition()** de **geolocation** permite saber la posición del usuario.

```

!DOCTYPE html>
<html lang="es-es">
<head>
 <title>Geolocation</title>
 <meta charset="UTF-8">
 <style>
 body { background-color: #ccffcc; }
 </style>
</head>
<body>
<p>Pulsa el botón para saber tus coordenadas</p>

<button onclick="coordenadas()">Pulsa</button>

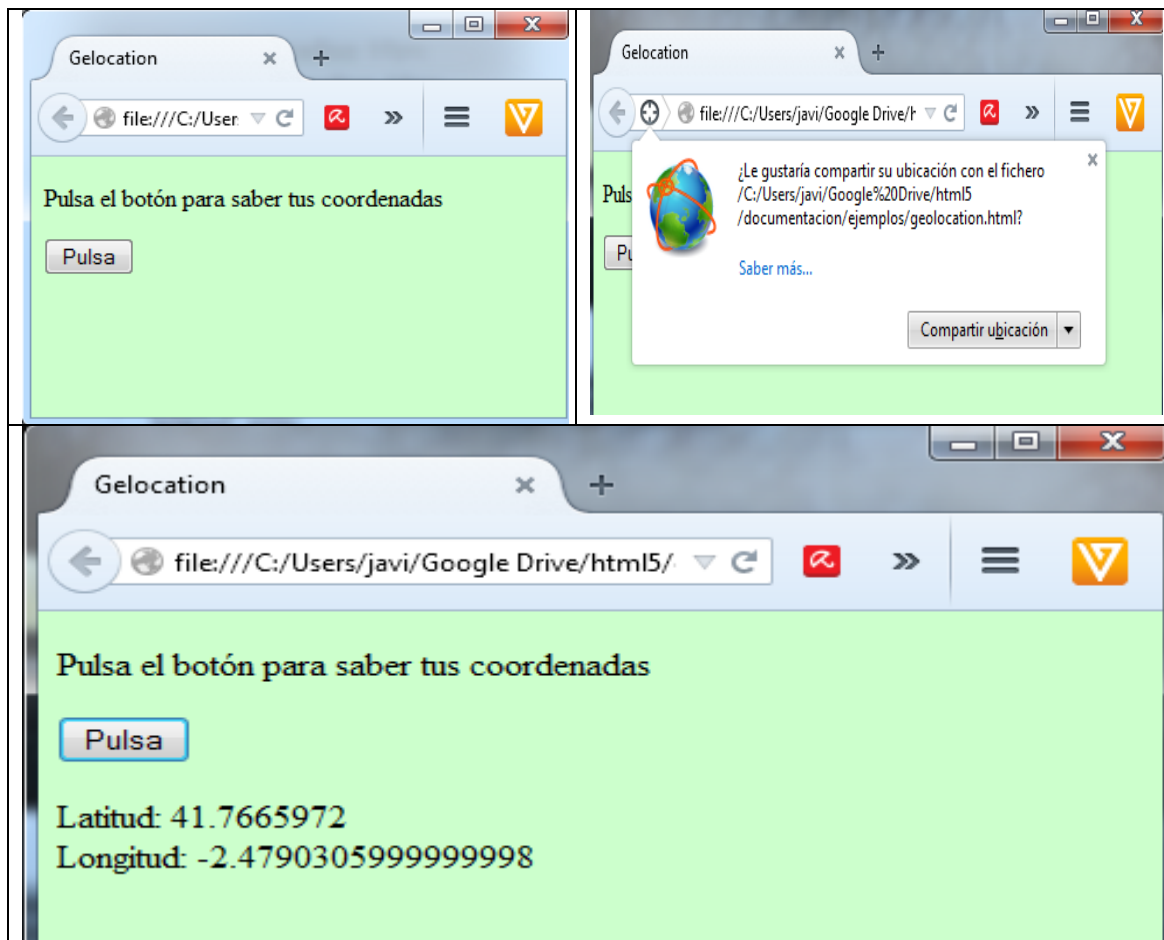
<p id="párrafo"></p>

<script>
var elemento = document.getElementById("párrafo");

function coordenadas() {
 if (navigator.geolocation) {
 navigator.geolocation.getCurrentPosition(posición);
 } else {
 elemento.innerHTML = "Geolocation no es soportada por tu navegador";
 }
}

function posición(lugar) {
 elemento.innerHTML = "Latitud: " + lugar.coords.latitude +
 "
Longitud: " + lugar.coords.longitude;
}
</script>
</body>
</html>

```



Ahora podemos modificar la página anterior y usar google maps para visualizar nuestra posición en un mapa

```
<!DOCTYPE html>
<html lang="es-es">
<head>
 <title>Gelocation con google</title>
 <meta charset="UTF-8">
 <style>
 body { background-color: #ccffcc; }
 </style>
</head>
<body>
<p>Pulsar el botón para saber tus coordenadas</p>

<button onclick="coordenadas()">Pulsar</button>

<section id="contenedor"></section>
<p id="párrafo"></p>

<script>
var elemento = document.getElementById("párrafo");

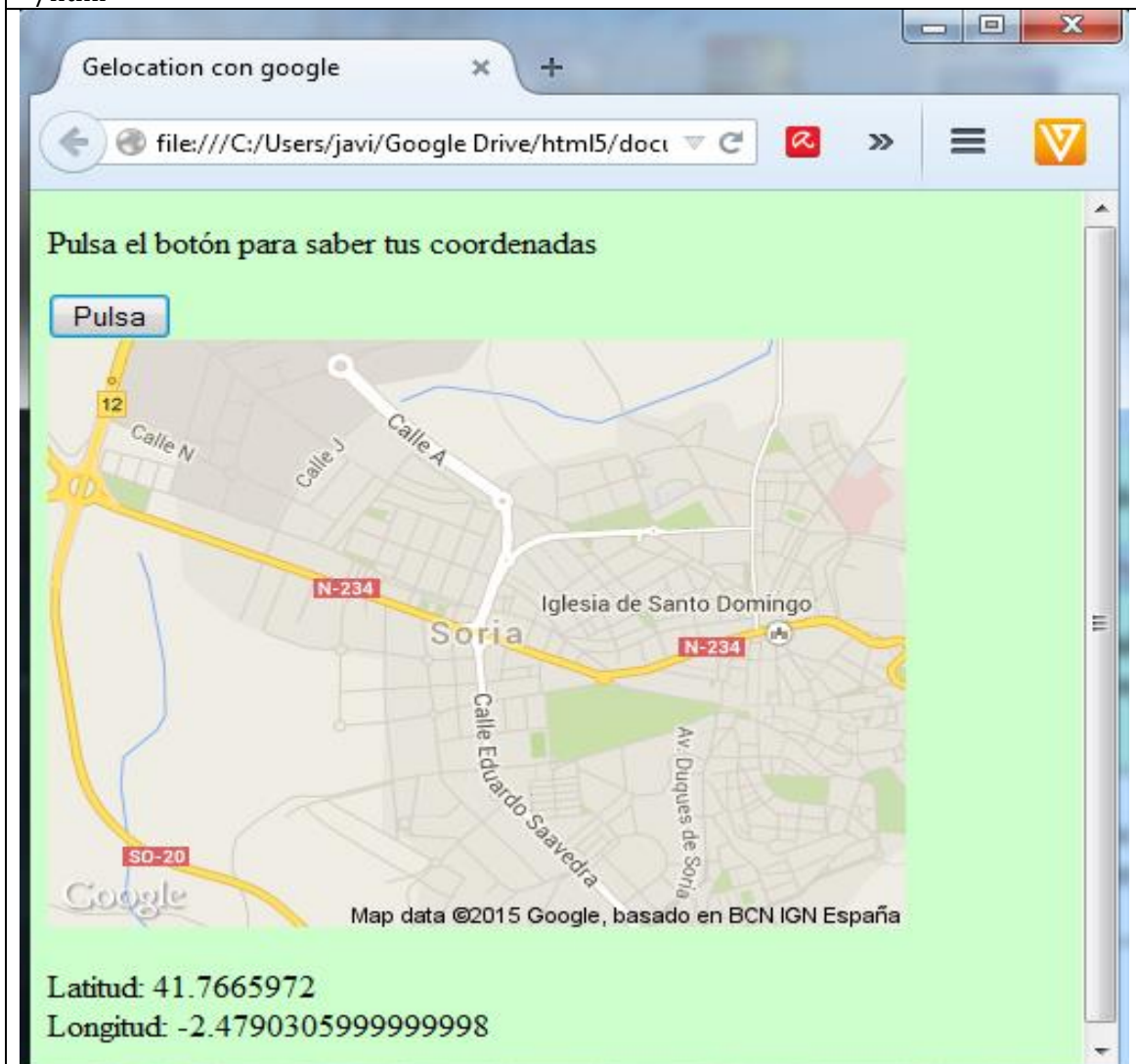
function coordenadas() {
 if (navigator.geolocation) {
```

```

 navigator.geolocation.getCurrentPosition(posición);
 } else {
 elemento.innerHTML = "Geolocation no es soportada por tu navegador";
 }
}

function posición(lugar) {
 elemento.innerHTML = "Latitud: " + lugar.coords.latitude + "
Longitud: " +
 lugar.coords.longitude;
 var mapa=lugar.coords.latitude + "," + lugar.coords.longitude;
 var
 imagenmapa="http://maps.googleapis.com/maps/api/staticmap?center="
 +mapa+"&zoom=14&size=400x300&sensor=false";
 document.getElementById("contenedor").innerHTML = "";
}
</script>
</body>
</html>

```



## 8.3. DRAG AND DROP

Esta API permite “arrastar y soltar” (**drag and drop**) un objeto para llevarlo de un lugar a otro dentro de la misma ventana o entre ventanas. Normalmente esta operación consta de tres partes:

- Se posiciona el ratón encima del objeto y se pulsa, normalmente, el botón izquierdo por lo que el objeto queda “agarrado”.
- Se arrastra desde la posición inicial en la que estaba dicho objeto hasta el lugar de destino.
- Se suelta el objeto en el lugar de destino.

API					
<b>Drag and Drop</b>	9.0	5.0	3.5	6.0	12.0

	<i>El API <b>Drag and Drop</b> forma parte del standard HTML5</i>
------------------------------------------------------------------------------------	-------------------------------------------------------------------

Para que a un elemento HTML5 se le pueda aplicar esta API, es decir, hacerlo “arrastrable” hay que agregar, en primer lugar, el atributo **draggable** con el valor **true**.

```

```

Ejemplo:

```
<!DOCTYPE HTML>

<html>
<head>
<style>
body { background-color: #ccffcc; }
section {
 width:700px;
 height:300px;
 border: 2px solid #ccddee;
 background-image: url("tablero.png");
```



```
 background-repeat: no-repeat;

 background-size: 700px 600px;

 background-position: center;
 }
img { display: inline;

 width: 88px;

 height: 73px;

 }

</style>
<script>
function permitir(evento) {
 evento.preventDefault();
}

function arrastrar(evento) {
 evento.dataTransfer.setData("text", evento.target.id);
}

function soltar(evento) {
 evento.preventDefault();

 var data = evento.dataTransfer.getData("text");

 evento.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>
```

```
<h1> Deja la torre y el caballo en el tablero</h1>
```

```
<section ondrop="soltar(event)" ondragover="permitir(event)"></section>
```

```


```

```

```

```

```

```
</body>
```

```
</html>
```





En este ejemplo partimos de dos imágenes (una torre y un caballo del juego del ajedrez) que están debajo de un tablero de ajedrez. Utilizando dos operaciones “drag and drop”

colocamos primero el caballo y luego la torre dentro del tablero que es la imagen de fondo de un elemento `<section>`.

Para entender mejor el ejemplo y comprender el fundamento del evento “arrastrar y soltar” vamos a explicarlo por partes o pasos:

- *Hacer las imágenes de la torre y del caballo “arrastrables”:* Se agrega a los elementos `<img>` el atributo **draggable** con el valor “true”

```



```

- *Se especifica lo que va a suceder cuando se inicia la operación de arrastre de cualquiera de las dos imágenes:* El atributo **ondragstart** llama a la función **arrastrar()** que especifica qué elemento se puede arrastrar y el método **dataTransfer.setData()** define el tipo y la identificación del elemento arrastrado. En este caso, el tipo de datos es “texto” (text) y el valor del id es “torre” y “caballo”.

```
function arrastrar(evento) {
 evento.dataTransfer.setData("text", evento.target.id);
}


```

- *Dónde dejar el elemento que es arrastrado:* El evento **ondragover** especifica donde puede ser dejado el elemento que es arrastrado.

Por defecto, no se puede dejar un elemento arrastrado dentro de otro elemento, por lo que, si queremos hacer esto, debemos emplear el método **preventDefault()** del evento **ondragover**.

```
function permitir(evento) {
 evento.preventDefault();
}

<section ondrop="soltar(event)" ondragover="permitir(event)"></section>
```

- *Soltar el elemento:* Cuando el elemento arrastrado es soltado se produce un evento “soltar” o “drop”. En el ejemplo se emplea la función “soltar”.

```
function soltar(evento) {
 evento.preventDefault();
 var data = evento.dataTransfer.getData("text");
 evento.target.appendChild(document.getElementById(data));
}

<section ondrop="soltar(event)" ondragover="permitir(event)"></section>
```

## 8.4. CANVAS

El nuevo elemento <canvas> de HTML5 permite dibujar gráficos de forma dinámica (que cambian en tiempo real) dentro de una página web a través de scripts escritos normalmente en javascript.

Etiqueta	<canvas>				
Descripción	Elemento para dibujar gráficos				
Elemento	Bloque				
Navegadores que la soportan					
	Chrome	Firefox	IE Explorer	Ópera	Safari
	4.0	2.0	9.0	9.0	3.1
Atributos	Globales y eventos				
Atributos propios	height, width				
Diferencia entre html 4.01 y html5	El elemento <canvas> es nuevo en HTML5				

	<i>Aunque &lt;canvas&gt; es un elemento también se emplea un API canvas para las operaciones de dibujo de gráficos</i>
------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------

### Ejemplo simple:

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
</body>
</html>
```



En este ejemplo simplemente hemos definido el elemento `<canvas>` con id igual a "dibujo" con una anchura de 500 pixeles, una altura de 300 pixeles y un borde de 3 pixeles de color marrón. Dentro de este elemento es donde vamos a "dibujar" a través de scripts javascript en los cuales se usan métodos de canvas.

#### Ejemplo de dibujo de un rectángulo:

	<i>Todos los dibujos canvas deben ser dibujados dentro de scripts javascript normalmente</i>
-------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border: 3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
```

Tu navegador no soporta <canvas>;

</canvas>

<script>

var gráfico = document.getElementById("dibujo");

var contexto = gráfico.getContext("2d");

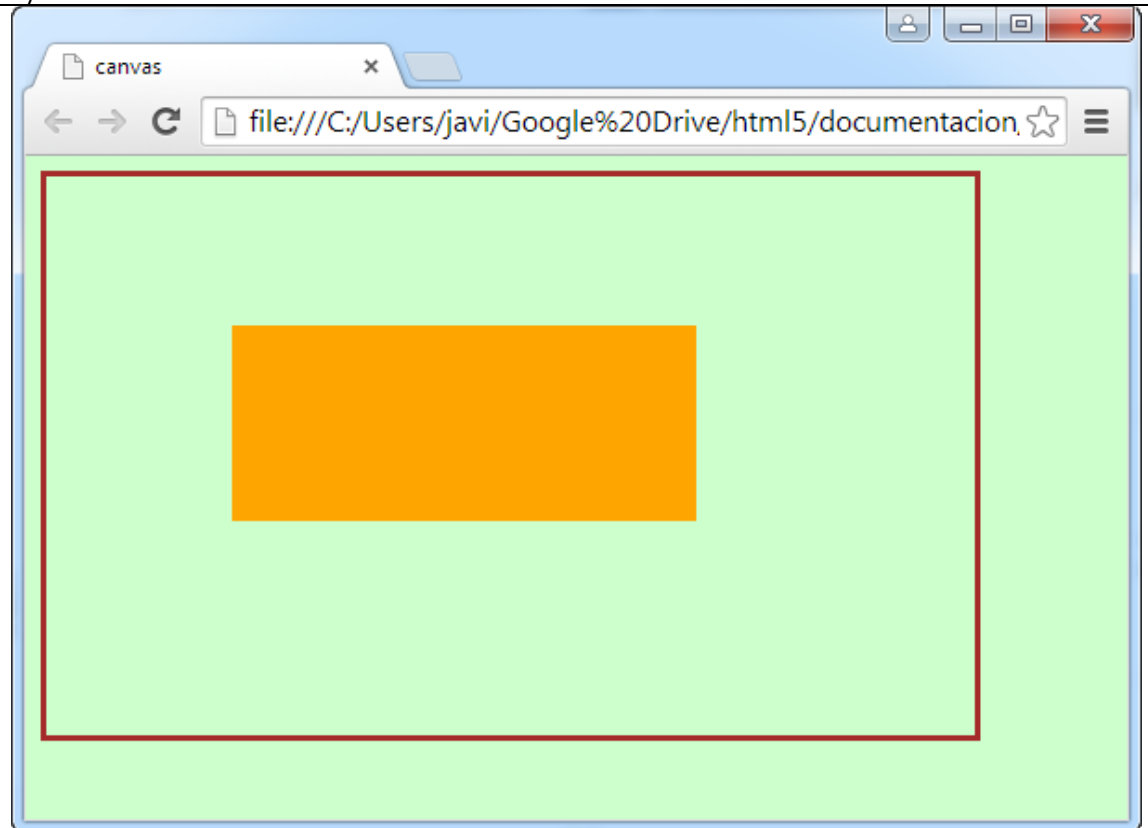
contexto.fillStyle = "orange";

contexto.fillRect(100,80,250,105);

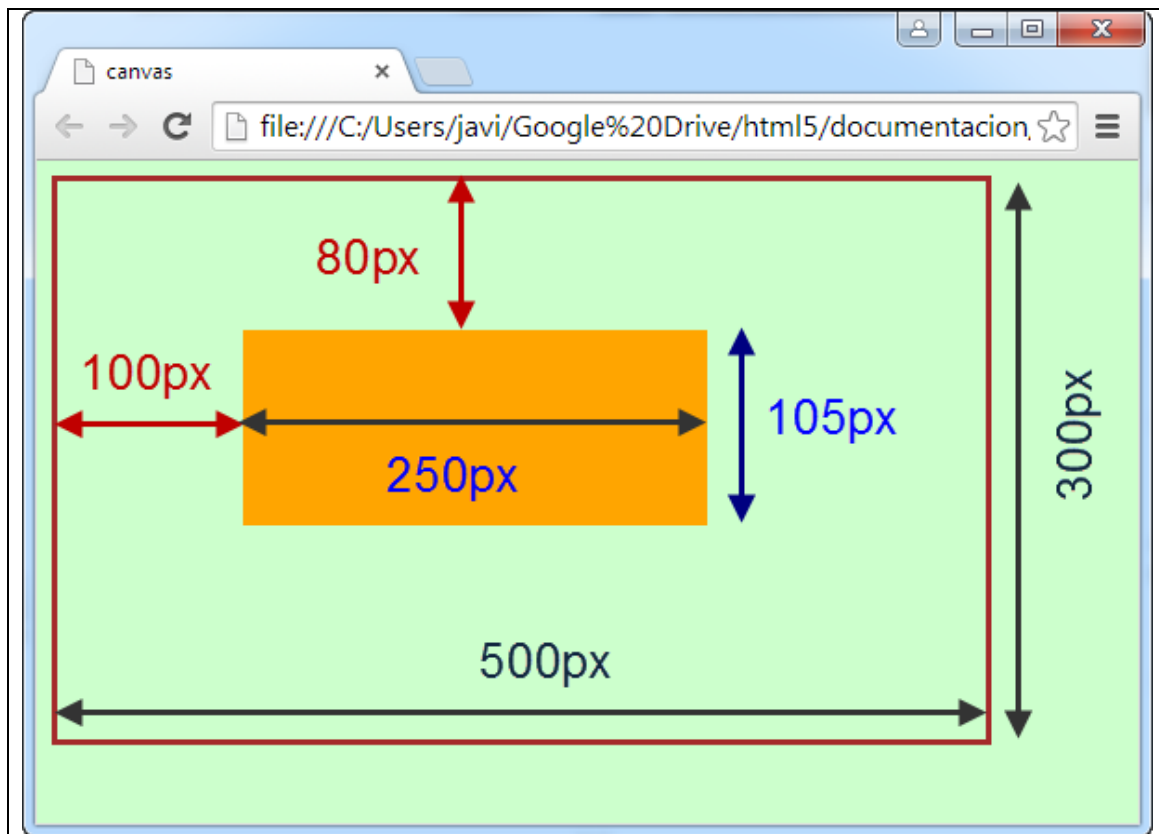
</script>

</body>

</html>







En este ejemplo creamos el elemento `<canvas>` con id igual a “dibujo” de 300 pixeles de alto por 500 de ancho que va a ser el área donde vamos a dibujar. A través de un script javascript se dibuja en el área anterior un rectángulo con relleno de color naranja (`contexto.fillStyle="orange"`) que comienza en el punto 100 (100 pixeles desde el borde izquierdo del elemento “dibujo”), 80 (80 pixeles desde el borde superior del elemento “dibujo”) con una anchura de 250 pixeles y una altura de 105 pixeles (`contexto.fillRect(100,80,250,105)`) siguiendo estos pasos:

- *Buscar el elemento `<canvas>` con id igual a “dibujo”*: Esto se hace usando el método HTML DOM `getElementById()` y asignándolo a la variable “gráfico”  
`var gráfico = document.getElementById("dibujo");`
- *Crear un objeto dibujable*: A través del método `getContext()` definimos el objeto “contexto” al que se puede aplicar propiedades y métodos de dibujo.  
`var contexto = gráfico.getContext("2d");`
- *Dibujar dentro del elemento `<canvas>` con id igual a “dibujo”*: Siguiendo el ejemplo primero le asignamos el color de relleno naranja al objeto “contexto” a través de la propiedad `fillStyle`.  
`contexto.fillStyle = "orange";`

Y, por último, dibujamos un rectángulo relleno con el color asignado a la propiedad `fillStyle` usando el método `fillRect()`.

`contexto.fillRect(100,80,250,105);`

### 8.4.1. Coordenadas canvas

Un elemento <canvas> es un cuadrilátero con dos dimensiones (ancho y alto). Lo que dibujamos dentro del elemento <canvas> tiene unas coordenadas que lo posicionan en un lugar concreto. La esquina superior izquierda del elemento <canvas> es el punto (0,0) donde la componente izquierda corresponde con la distancia del borde izquierdo del elemento <canvas> y la componente derecha es la distancia desde el borde superior. Si nos fijamos en la imagen superior vemos que el rectángulo tiene las coordenadas (100,80), es decir, esas coordenadas se corresponden con el punto desde el cual se empieza a dibujar la figura. El número de la izquierda (100) es la distancia al borde izquierdo y el de la derecha (80) es la distancia desde el borde superior.

### 8.4.2. Dibujando una línea recta

Para dibujar una línea recta dentro de un elemento <canvas> se utilizan los métodos:

- moveTo(izda, arriba): Define el punto inicial de la recta.
- lineTo(izda,arriba): Define el punto final de la recta.
- stroke(): Dibuja la línea

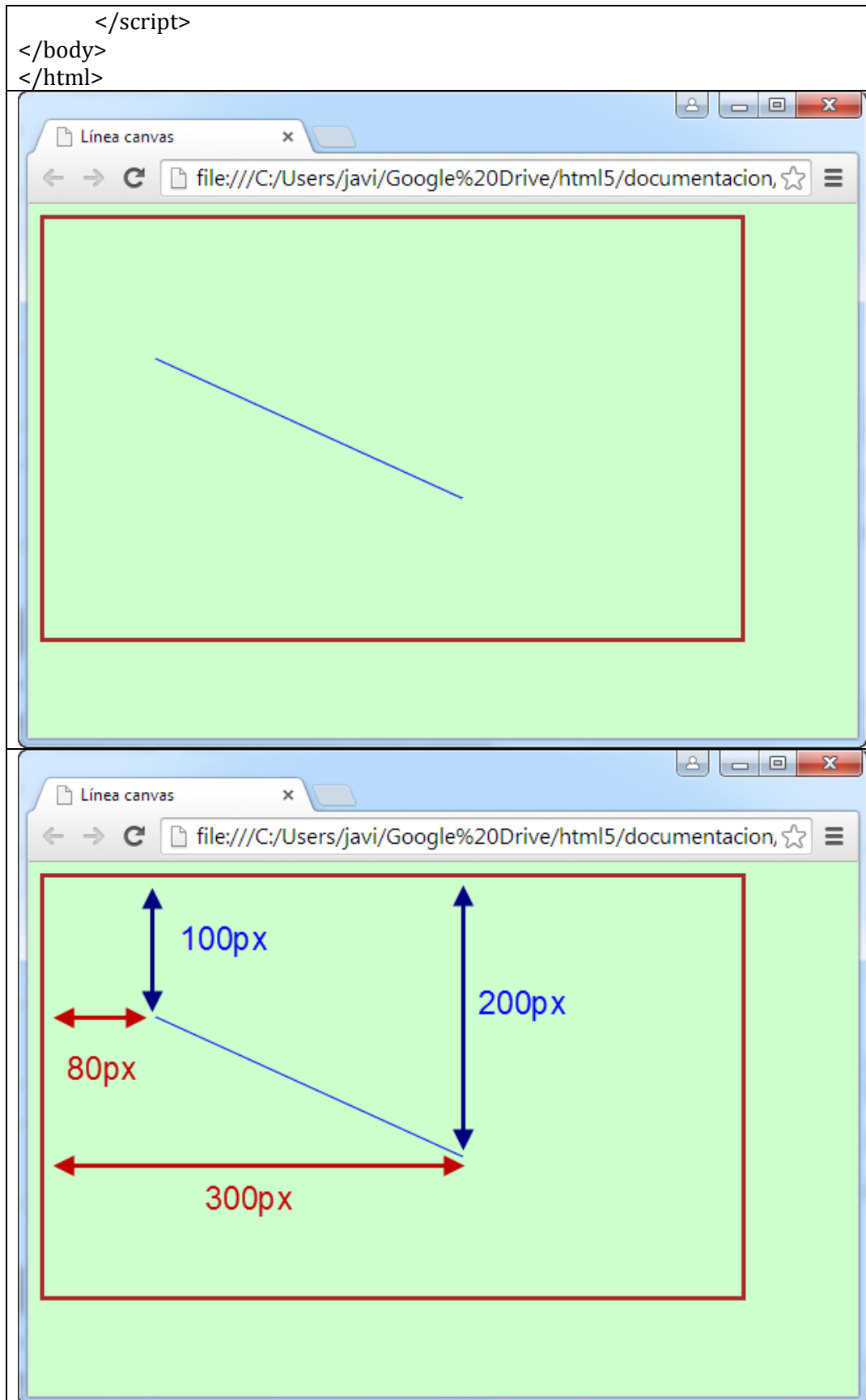
Y la propiedad:

- strokeStyle: Define el color de la línea.

#### Ejemplo:

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Línea canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
 var contexto = gráfico.getContext("2d");
 contexto.moveTo(80,100);
 contexto.lineTo(300,200);
 contexto.strokeStyle="blue";
 contexto.stroke();
```



Como vemos en este ejemplo se define el elemento `<canvas>` y, dentro ya del `<script>`, se usan las dos mismas declaraciones utilizadas para dibujar el

rectángulo anterior ya que hay que identificar el elemento <canvas> y hacerlo dibujable. Estas declaraciones son:

```
var gráfico = document.getElementById("dibujo");
var contexto = gráfico.getContext("2d");
```

Las siguientes declaraciones del script definen el punto inicial de la línea (contexto.moveTo(80,100);), el punto final de la misma (contexto.lineTo(300,200);), su color (contexto.strokeStyle="blue;") y se dibuja la recta (contexto.stroke();).

### Ejemplo de dibujo de varias líneas en un elemento <canvas>:

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Varias línea canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
 var contexto = gráfico.getContext("2d");

 // Líneas verticales

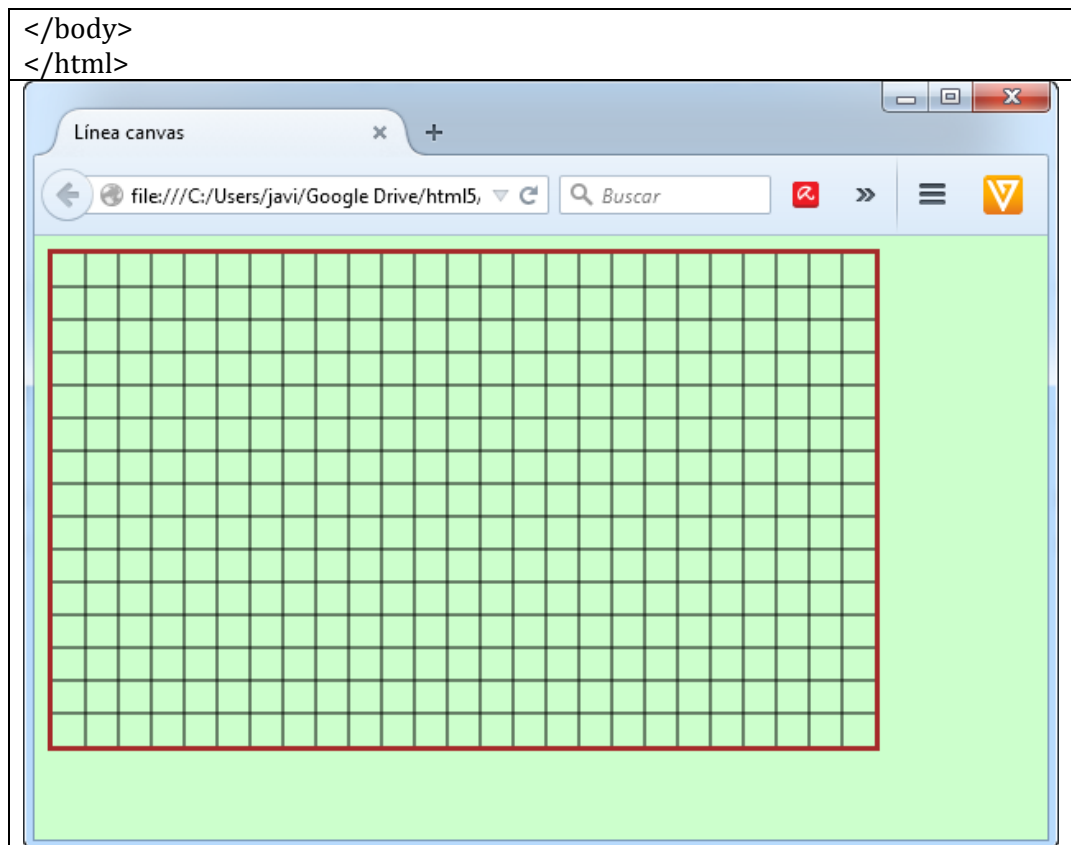
 for (var x = 20; x < 500; x += 20) {
 contexto.moveTo(x, 0);
 contexto.lineTo(x, 300);
 }

 //líneas horizontales

 for (var y = 20; y < 300; y += 20) {
 contexto.moveTo(0, y);
 contexto.lineTo(500, y);
 }

 //dibujo de todas las líneas en color negro

 contexto.strokeStyle = "black";
 contexto.stroke();
 </script>
```



En este ejemplo dibujamos un “grid” de líneas rectas tanto verticales como horizontales a través de dos bucles que separa las líneas rectas 20 pixeles entre sí. Como se ve sólo hace falta ejecutar una vez la propiedad `strokeStyle` y el método `stroke()` para que dibuje todas las líneas. Esto es debido a que, en realidad, “contexto” es un objeto y actúa como tal. Recordad también que las líneas que empiezan por “\” son comentarios en javascript.

#### 8.4.3. Dibujando un círculo

Para dibujar un círculo en canvas, además del método `stroke()` y de la propiedad `strokeStyle` vistos en el punto anterior, se usan los siguientes métodos:

- `beginPath()`: Indica que se va a dibujar una figura que puede cerrarse o no como, por ejemplo, polígonos o círculos o parte de ellos.
- `arc(izda,dcha,radio,ángulo inicial, ángulo final, sentido)`: Sirve para definir un arco con centro en las coordenadas (izda, dcha) con un radio “radio”, empezando en el “ángulo inicial”, acabando en el “ángulo final” y si el sentido del dibujo es igual al de las agujas del reloj (`true`) o al revés (`false`).

Y la propiedad:

- `lineWidth`: Se utiliza para indicar el grosor de la línea.

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Círculo canvas</title>
```

```

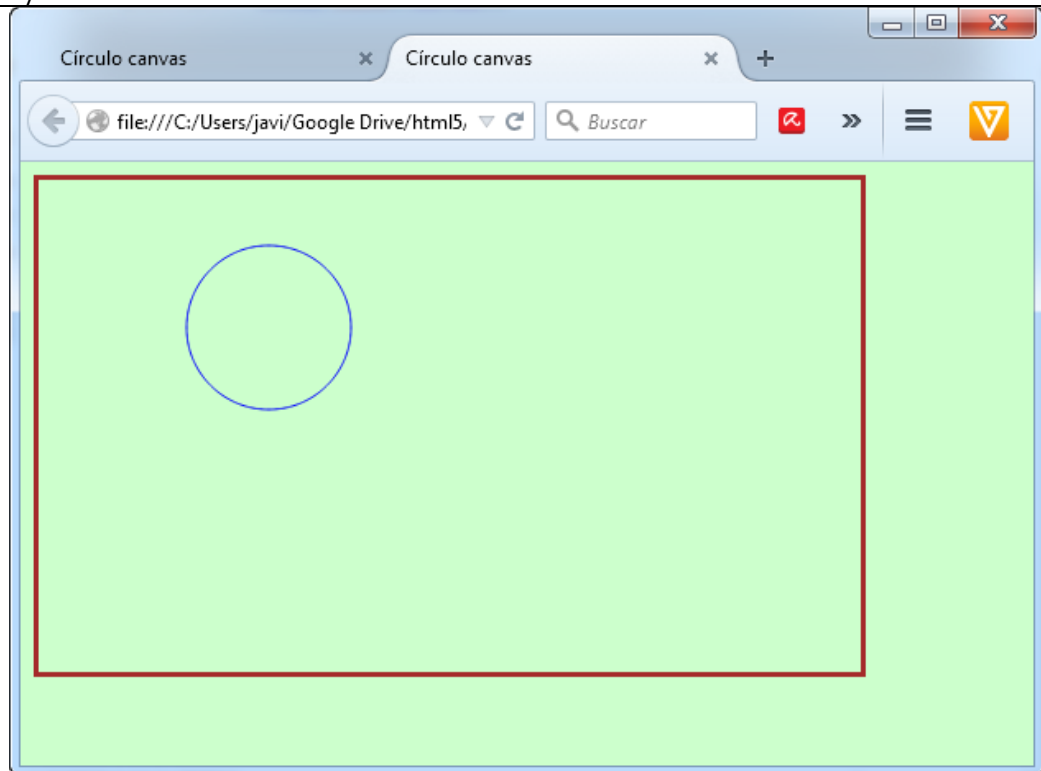
<style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
</style>

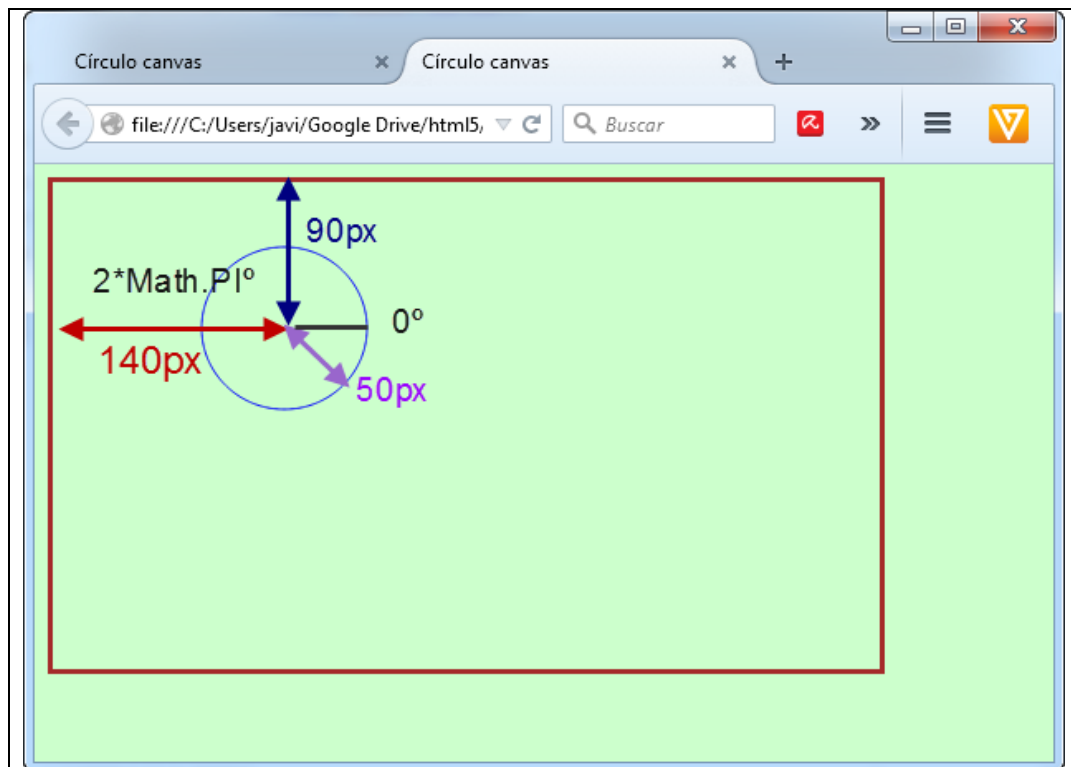
</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
 var círculo = gráfico.getContext("2d");

 círculo.beginPath();
 círculo.arc(140,90,50,0,2*Math.PI);
 círculo.strokeStyle = "blue";
 círculo.stroke();

 </script>
</body>
</html>

```





**Ejemplo que muestra varios dibujos con el método *arc***

```

<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Círculo canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
 var círculo = gráfico.getContext("2d");
 var círculo1 = gráfico.getContext("2d");
 var círculo2 = gráfico.getContext("2d");
 var círculo3 = gráfico.getContext("2d");
 var círculo4 = gráfico.getContext("2d");

 círculo.beginPath();
 círculo.arc(140,90,50,0,2*Math.PI);
 círculo.strokeStyle = "blue";
 </script>

```

```

círculo.stroke();

círculo1.beginPath();
círculo1.arc(350,90,50,0,Math.PI);
círculo1.strokeStyle = "blue";
círculo1.stroke();

círculo2.beginPath();
círculo2.arc(140,225,50,Math.PI,2*Math.PI);
círculo2.strokeStyle = "blue";
círculo2.stroke();

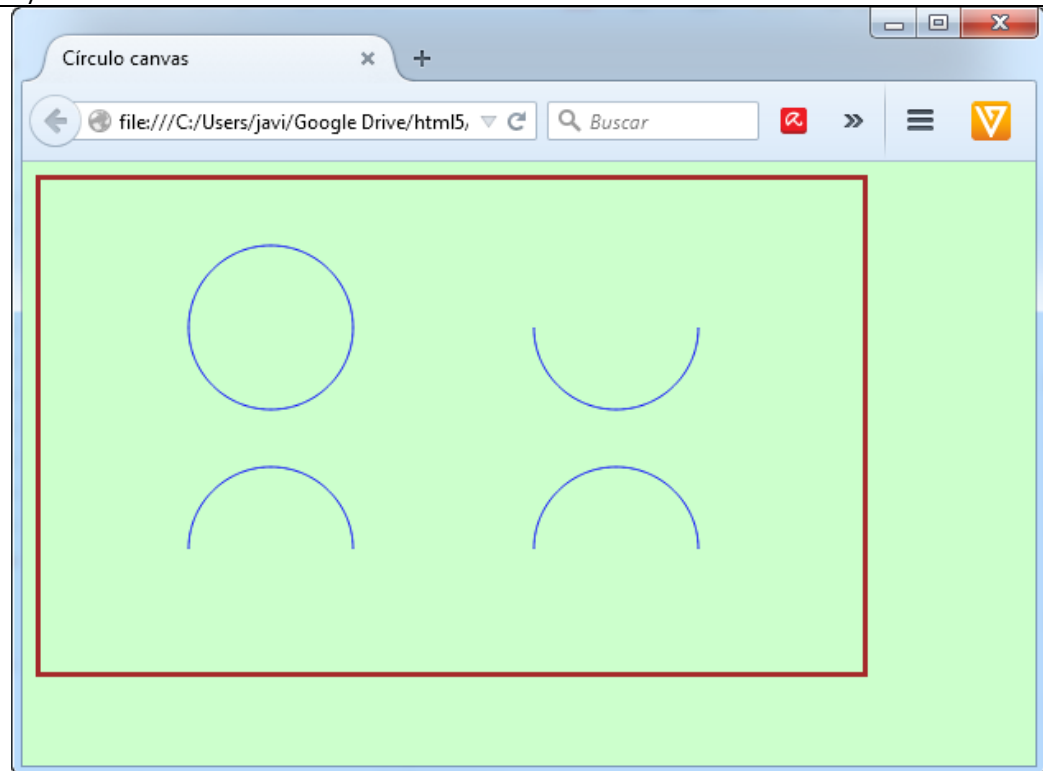
círculo3.beginPath();
círculo3.arc(350,225,50,0,Math.PI,true);
círculo3.strokeStyle = "blue";
círculo3.stroke();

```

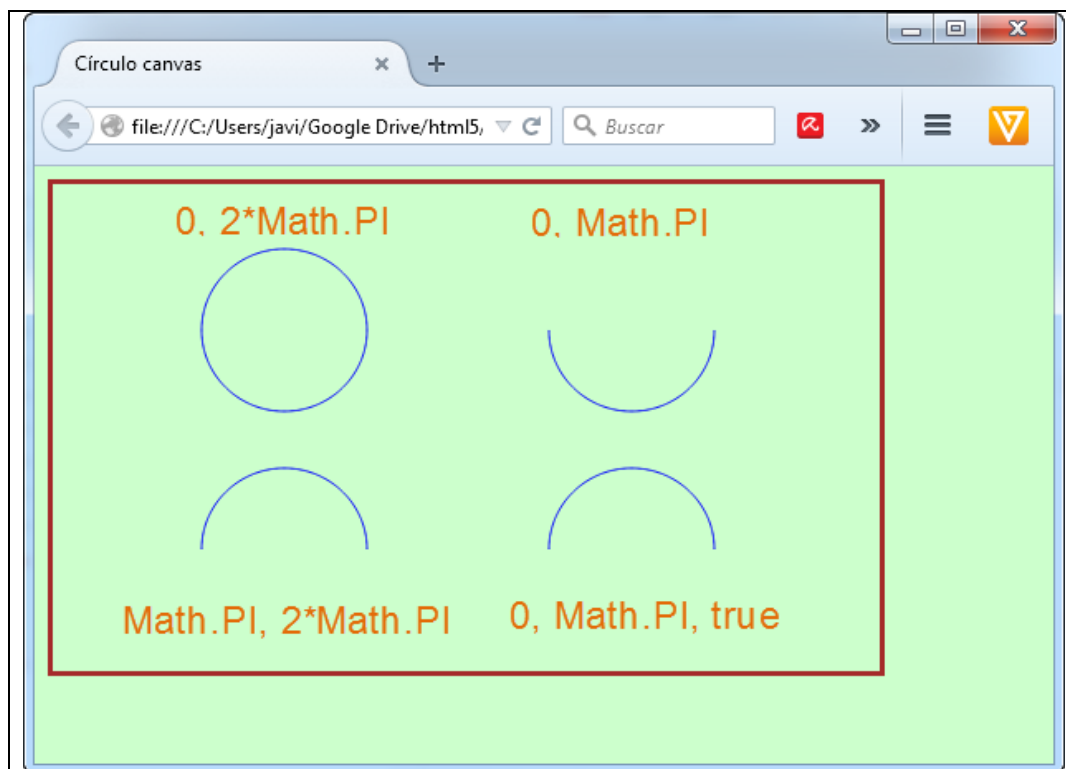
```

</script>
</body>
</html>

```







En estos ejemplos se emplea la constante “Math.PI” para la asignación de ángulos que se corresponde con el número pi (3,1415....) representado en radianes perteneciente al objeto predefinido “Math” de javascript. Este objeto permite operaciones matemáticas como redondeos, números aleatorios, etc.

#### 8.4.4. Gradientes o difuminados

En canvas se pueden rellenar textos o figuras como círculos, rectángulos, etc., de una determinada zona del elemento <canvas> con tonos difuminados o gradientes a través de los siguientes métodos:

- `createLinearGradient(izda,arriba,izda1, arriba1)`: Habilita una zona del elemento <canvas> donde se puede aplicar un difuminado lineal que empieza en el punto (izda, arriba) y acaba en el punto (izda1,arriba1).
- `createLinearGradient(izda,arriba,radio,izda1,arriba1,radio1)`: Habilita una zona del elemento <canvas> donde se puede aplicar un difuminado concéntrico que empieza en el punto (izda,arriba) con radio “radio” y acaba en el punto (izda1, arriba1) con radio “radio1”.
- `addColorStop(número de orden, color)`: Especifica el color que se difumina y el orden dentro del relleno. El “número de orden” estará comprendido entre 0 y 1 pudiéndose emplear números decimales como, por ejemplo, 0.5.

Para emplear los gradientes en una figura se asigna el objeto que se ha creado como gradiente a las propiedades `fillStyle` o `strokeStyle`.

**Ejemplo de difuminado lineal de izquierda a derecha de una zona donde se dibuja un rectángulo:**

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Rectángulo difuminado canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
 var rectángulo = gráfico.getContext("2d");

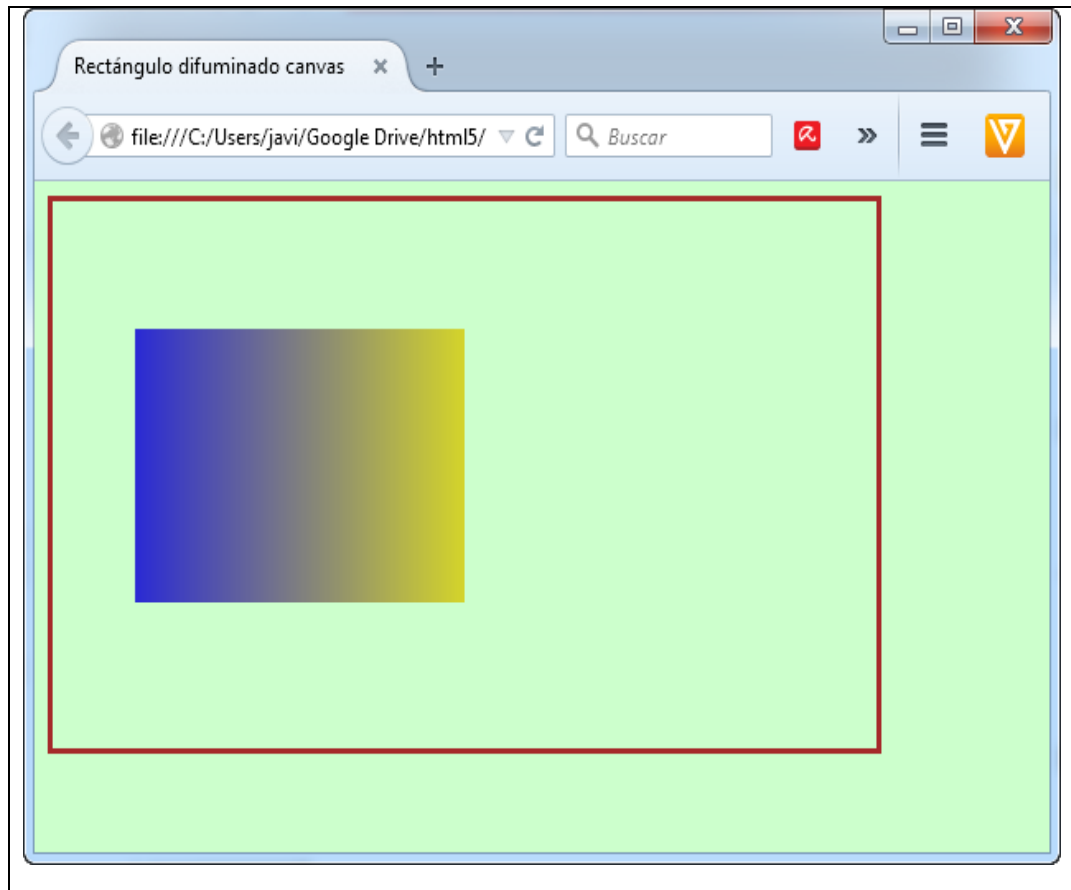
 //creando difuminado

 var difuminado=rectángulo.createLinearGradient(0,0,300,0);
 difuminado.addColorStop (0,"blue");
 difuminado.addColorStop (1,"yellow");

 //usando difuminado

 rectángulo.fillStyle= difuminado;
 rectángulo.fillRect (50,100,200,150);

 </script>
</body>
</html>
```



En este ejemplo se define un difuminado de izquierda a derecha en la zona que va desde el borde izquierdo hasta 300 puntos de ancho que cambia de color azul a amarillo y dentro de esa zona se dibuja un rectángulo al cual se aplica el difuminado.

**Ejemplo de difuminado lineal de arriba a abajo de una zona donde se dibuja un rectángulo:**

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Rectángulo difuminado canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
```

```

Tu navegador no soporta <canvas>
</canvas>
<script>
 var gráfico = document.getElementById("dibujo");
 var rectángulo = gráfico.getContext("2d");

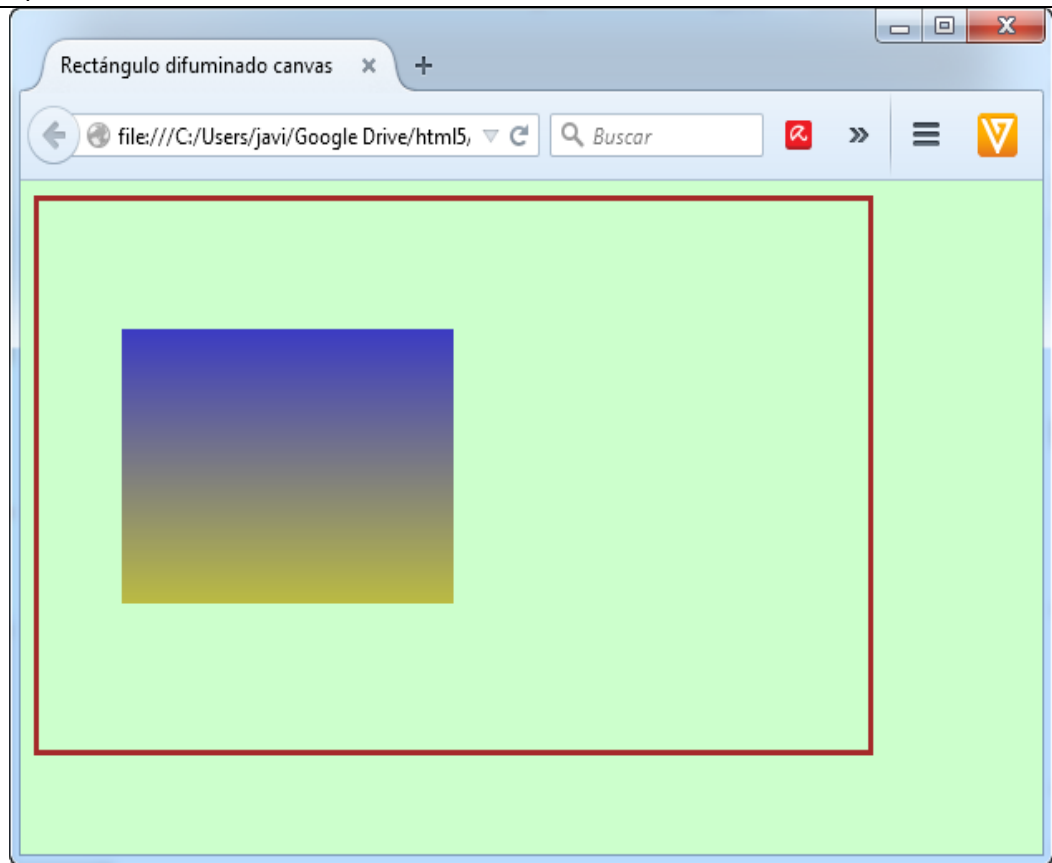
 //creando difuminado
 var difuminado=rectángulo.createLinearGradient(0,0,0,300);
 difuminado.addColorStop (0,"blue");
 difuminado.addColorStop (1,"yellow");

 //usando difuminado

 rectángulo.fillStyle= difuminado;
 rectángulo.fillRect (50,70,200,150);

</script>
</body>
</html>

```



En este ejemplo se define un difuminado de arriba a abajo en la zona que va desde el borde superior hasta 300 puntos más abajo que cambia de color azul a amarillo y dentro de esa zona se dibuja un rectángulo al cual se aplica el difuminado.

**Ejemplo de difuminado radial de una zona donde se va a dibujar un rectángulo:**

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Rectángulo difuminado canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

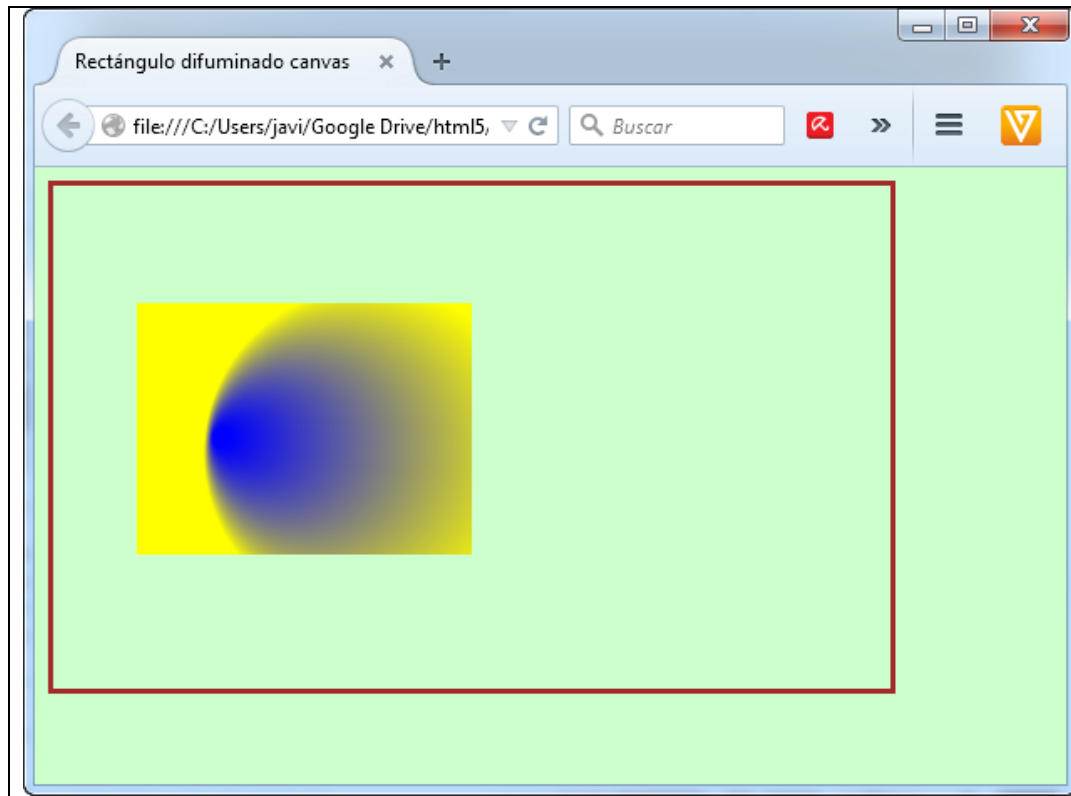
</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
 var rectángulo = gráfico.getContext("2d");

 //creando difuminado
 var
difuminado=rectángulo.createRadialGradient(100,150,5,190,160,100);
 difuminado.addColorStop (0,"blue");
 difuminado.addColorStop (1,"yellow");

 //usando difuminado

 rectángulo.fillStyle= difuminado;
 rectángulo.fillRect (50,70,200,150);

 </script>
</body>
</html>
```



#### 8.4.5. Texto

Los métodos y propiedades más importantes que se emplean para dibujar textos en canvas son:

- font: Define las propiedades de la fuente del texto tales como tamaño, familia, etc.
- fillText(texto, izda, arriba): Dibuja texto con relleno donde el par de números izda y arriba representan las coordenadas de donde se va a empezar a escribir.
- strokeText(texto, izda, arriba): Dibuja texto sin relleno. Al igual que fillText() izda y arriba son las coordenadas de donde empieza el texto.
- textAlign: Alinea el texto.

#### Ejemplo de texto con relleno:

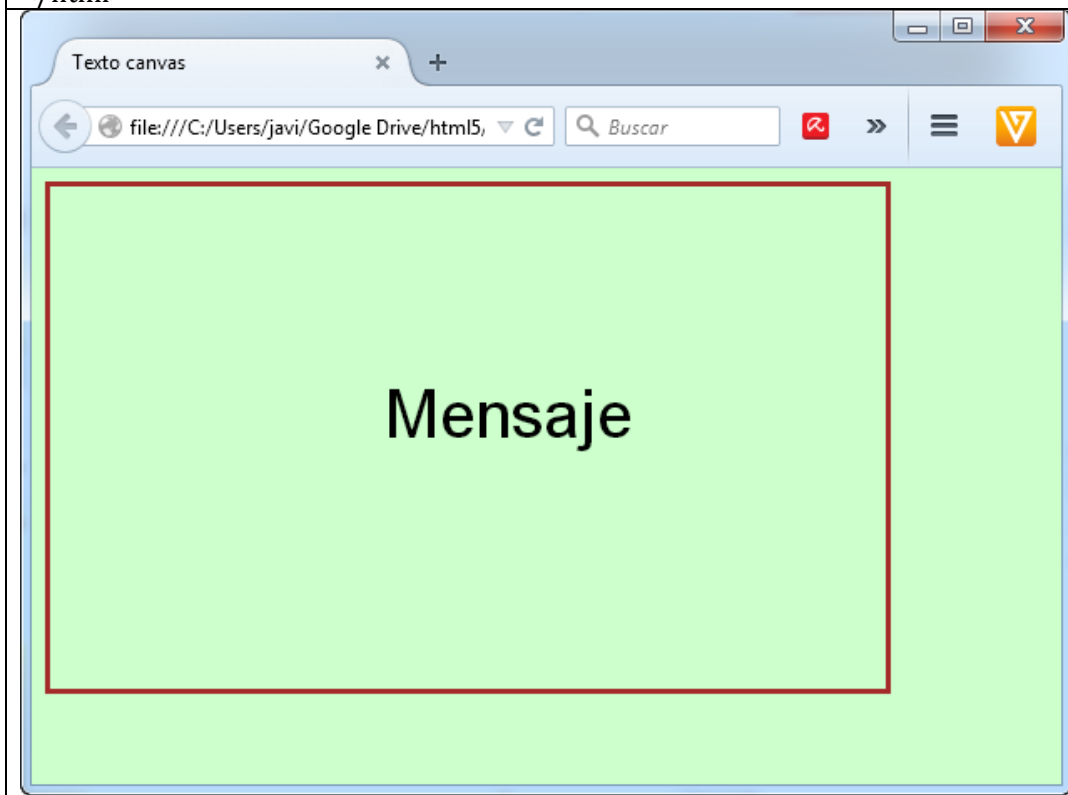
```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Texto canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border: 3px solid brown;
 }
 </style>
</head>
<body>
 <div id="dibujo">

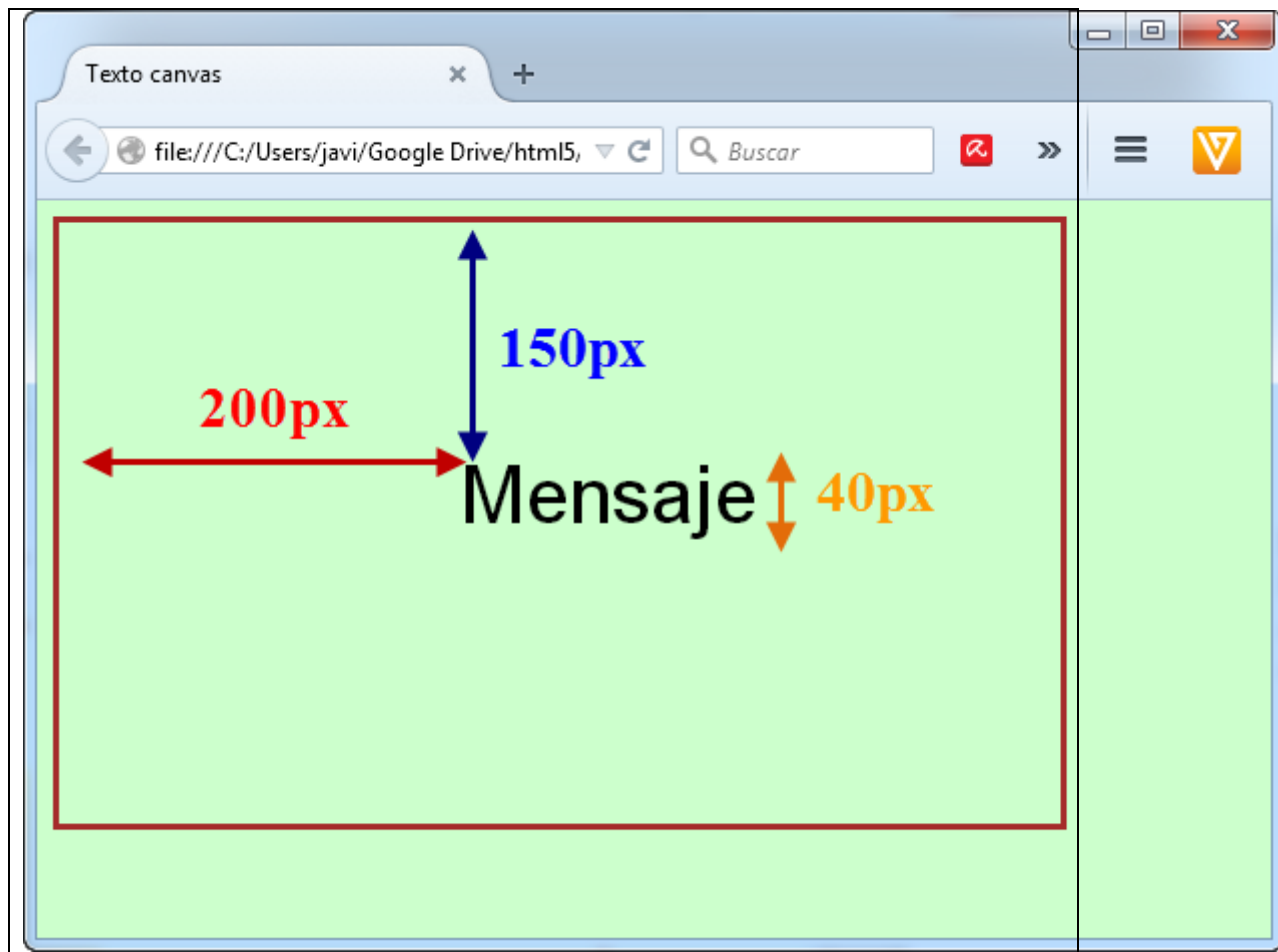
 </div>
</body>
```

```
</style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
 var texto = gráfico.getContext("2d");
 texto.font = "40px Arial";
 texto.fillText("Mensaje", 200,150);

 </script>
</body>
</html>
```





### Ejemplo de texto con letra hueca:

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Texto canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

</head>
<body>
 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>
 </canvas>
 <script>
 var gráfico = document.getElementById("dibujo");
```

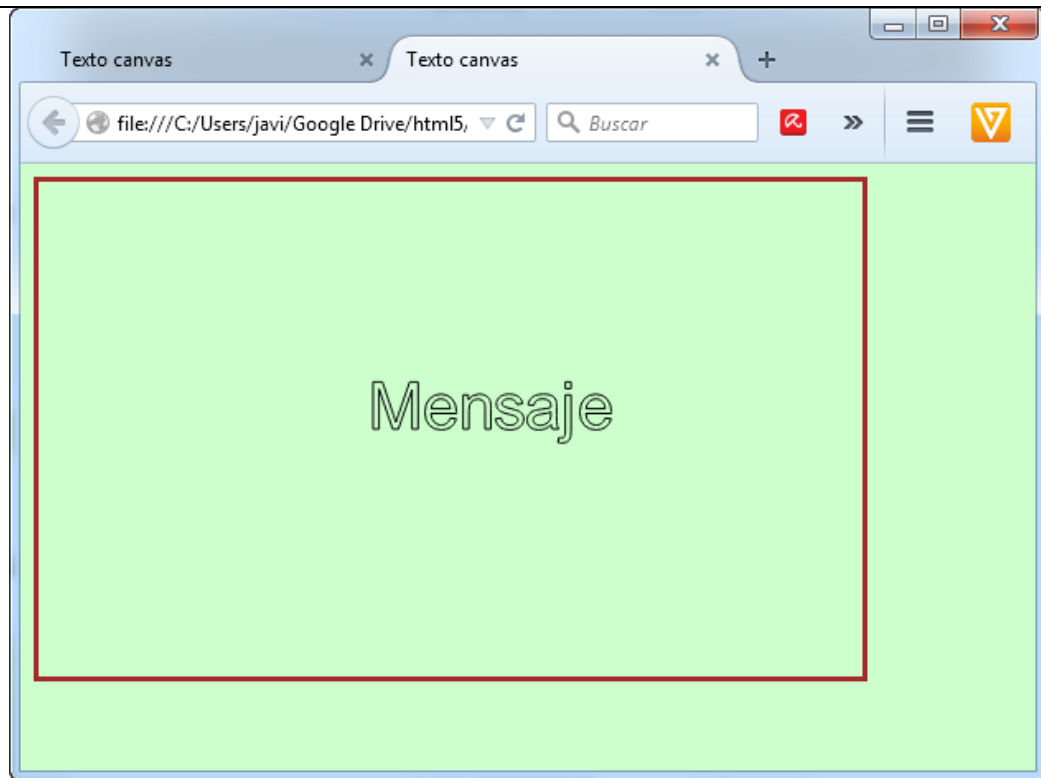


```

 var texto = gráfico.getContext("2d");
 texto.font = "40px Arial";
 texto.strokeText("Mensaje", 200,150);

 </script>
</body>
</html>

```



### Ejemplo de texto centrado y de color azul:

```

<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Texto canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>

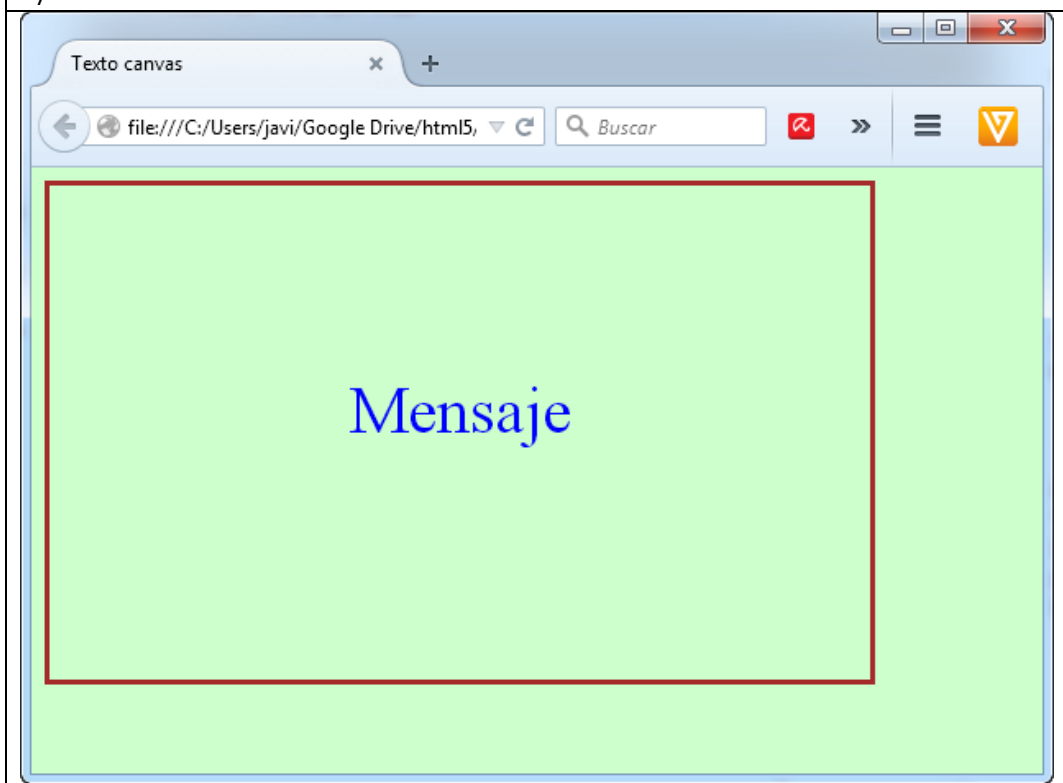
</head>
<body>
 <canvas id="dibujo" height="300" width="500">

```

Tu navegador no soporta <canvas>;

```
</canvas>
<script>
 var gráfico = document.getElementById("dibujo");
 var texto = gráfico.getContext("2d");
 texto.font = "40px Times New Roman";
 texto.fillStyle = "blue";
 texto.textAlign = "center";
 texto.fillText("Mensaje", gráfico.width/2, gráfico.height/2);

</script>
</body>
</html>
```



#### 8.4.6. Imágenes

Para insertar una imagen en un elemento <canvas> se utiliza el siguiente método:

- drawImage(imagen,izda,arriba): Dibuja la imagen "imagen" en la posición "izda", "arriba".



*No se puede dibujar una imagen antes de que dicha imagen se haya cargado por lo que hay que usar una función que se ejecute cuando se carga la página como puede ser **window.onload()***

**Ejemplo:**

```
<!DOCTYPE html>
<html lang="es-es"
<head>
 <meta charset="UTF-8">
 <title>Imágenes canvas</title>
 <style>
 body { background-color: #ccffcc; }
 #dibujo {
 border:3px solid brown;
 }
 </style>
</head>
<body>

<h2>Imagen a usar en canvas</h2>

<h2>Canvas</h2>

 <canvas id="dibujo" height="300" width="500">
 Tu navegador no soporta <canvas>;
 </canvas>

<script>
window.onload = function() {
 var gráfico = document.getElementById("dibujo");
 var imagen = gráfico.getContext("2d");
 var img = document.getElementById("icono");
 imagen.drawImage(img, 200, 50);
}
</script>

</body>
</html>
```

